Self-adaptive Differential Evolution with Neighborhood Search

Zhenyu Yang, Ke Tang and Xin Yao

***Abstract*— In this paper we investigate several self-adaptive mechanisms to improve our previous work on NSDE [1], which is a recent DE variant for numerical optimization. The self-adaptive methods originate from another DE variant, SaDE [2], but are remarkably modified and extended to fit our NSDE. And thus a Self-adaptive NSDE (SaNSDE) is proposed to improve NSDE's performance. Three self-adaptive mechanisms are utilized in SaNSDE: self-adaptation for two candidate mutation strategies, self-adaptations for controlling scale factor F and crossover rate CR, respectively. Experimental studies are carried out on a broad range of different benchmark functions, and the proposed SaNSDE has shown significant superiority over NSDE.**

I. Introduction

DIFFERENTIAL evolution (DE) [3] has become a popular algorithm in global optimization. It has shown superior performance in both widely used benchmark functions [4], [5] and real-world applications [6]. DE conventionally has several candidate mutation schemes, and three control parameters, i.e., population size NP, scale factor F and crossover rate CR. Apart from the parameter NP (which is common for all population-based algorithms), mutation strategy selection, parameters F and CR adaptations are the three most important issues of DE research. Many work has been done along these lines. The relationship between the control parameters and population diversity has been analyzed in [7]. Experimental parameter studies and empirical parameter settings of DE have been carried out in [8]. Self-adaptive strategy has also been investigated to adapt these control parameters [9], as well as different mutation strategies [2].

In [1], we proposed a DE variant, namely Differential Evolution with Neighborhood Search (NSDE), to adapt the scale factor F. Inspired by the neighborhood search (NS) strategy in evolutionary programming (EP) [10], NSDE intends to mix search biases of different NS operators through the factor F. It is well-known that NS is a main strategy underpinning EP [11]. Although DE might be similar to the evolutionary process of EP, it lacks relevant concept of neighborhood search. Instead of predefining the factor F as a constant, NSDE generates F from Gaussian and Cauchy distributed random numbers, which are beneficial to producing small and large search step sizes, respectively [11]. A probability fp is introduced to control when to use Gaussian or Cauchy operator. In the previous work fp was simply set to a constant number 0.5. Obviously, it would be more desirable if fp could be self-adapted during the evolution process. By these means, the algorithm can automatically adjust between Gaussian and Cauchy operators, and thereby the performance can be improved.

Self-adaptive Differential Evolution (SaDE) by Qin et at. [2], is a different DE variant that mainly focuses on adaptation for parameter CR and mutation strategies of DE. The motivation is to solve the dilemma that CR values and mutation strategies involved in DE are often highly problem dependent. SaDE adopts two DE mutation strategies and introduces a probability p to control which one to use. The probability p is gradually self-adapted according to learning experience. Additionally, crossover rate CR is self-adapted by recording CR values that make offspring successfully enter the next generation. Both of the two self-adaptive mechanisms have achieved significant improvement over the classical DE with empirical parameter configuration.

It can be concluded that SaDE and NSDE have quite different emphases on improving DE's performance. SaDE pays special attention to the crossover rate CR's adaptation and the self-adaptation between different DE mutation strategies, while NSDE intends to mix search biases of different NS operators through the parameter F, and no self-adaptation is adopted. The difference motivates us to introduce SaDE's self-adaptive mechanisms into NSDE, study their behaviors, and then propose a self-adaptive NSDE (SaNSDE). The outline and features of the proposed SaNSDE are summarized as follows: 1) It inherits the self-adapted mutation schemes selection mechanism of SaDE; 2) It adopts a self-adaptive strategy to adjust the parameter fp of NSDE; 3) It enhances the original CR self-adaptation of SaDE by adding a weighting strategy. The efficacy of SaNSDE is evaluated on two sets of widely used benchmark functions.

The rest of this paper is organized as follows: Section II gives the preliminaries; Section III describes the proposed SaNSDE algorithm; Section IV presents the experimental studies; Finally, Section V concludes this paper and briefly discusses several other self-adaptive DE schemes.

II. Preliminaries

A. Differential Evolution (DE)

Individuals in DE are represented by D-dimensional vectors $\mathbf{x}_i, \forall i \in \{1, \cdots, NP\}$, where D is the number of objective parameters and NP is the population size. According to [3], the classical DE can be summarized as follows:

1) Mutation:

$$\mathbf{v}_i = \mathbf{x}_{i_1} + F \cdot (\mathbf{x}_{i_2} - \mathbf{x}_{i_3})$$

The authors are with the Nature Inspired Computation and Applications Laboratory, the Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China. Xin Yao is also with CERCIA, the School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. (emails: zhyuyang@mail.ustc.edu.cn, ketang@ustc.edu.cn, x.yao@cs.bham.ac.uk).

Corresponding author: Ke Tang (Phone: +86–551–3600754).

978-1-4244-1823-7/08/$25.00

where $i_1, i_2, i_3 \in [1, NP]$ are random and mutually different integers, and they are also different with the vector index i. Scale factor $F > 0$ is a real constant factor and is often set to 0.5.

2) Crossover:

$$\mathbf{u}_i(j) = \begin{cases} \mathbf{v}_i(j), & if\ U_j(0,1) \leq CR\ or\ j = j_{rand} \\ \mathbf{x}_i(j), & otherwise. \end{cases}$$

with $U_j(0,1)$ stands for the uniform random number between 0 and 1, and j_{rand} is a randomly chosen index to ensure that the trial vector $\mathbf{u}_i$ does not duplicate $\mathbf{x}_i$. $CR \in (0,1)$ is the crossover rate, which is often set to 0.9.

3) Selection:

$$\mathbf{x}'_i = \begin{cases} \mathbf{u}_i, & if\ f(\mathbf{u}_i) \leq f(\mathbf{x}_i) \\ \mathbf{x}_i, & otherwise. \end{cases}$$

where $\mathbf{x}'_i$ is the offspring of $\mathbf{x}_i$ for the next generation (Without loss of generality, we consider only minimization problem in this paper).

There are several schemes of DE based on different mutation strategies [3]:

$$\mathbf{v}_i = \mathbf{x}_{i_1} + F \cdot (\mathbf{x}_{i_2} - \mathbf{x}_{i_3}) \quad (1)$$

$$\mathbf{v}_i = \mathbf{x}_{best} + F \cdot (\mathbf{x}_{i_1} - \mathbf{x}_{i_2}) \quad (2)$$

$$\mathbf{v}_i = \mathbf{x}_i + F \cdot (\mathbf{x}_{best} - \mathbf{x}_i) + F \cdot (\mathbf{x}_{i_1} - \mathbf{x}_{i_2}) \quad (3)$$

$$\mathbf{v}_i = \mathbf{x}_{best} + F \cdot (\mathbf{x}_{i_1} - \mathbf{x}_{i_2}) + F \cdot (\mathbf{x}_{i_3} - \mathbf{x}_{i_4}) \quad (4)$$

$$\mathbf{v}_i = \mathbf{x}_{i_1} + F \cdot (\mathbf{x}_{i_2} - \mathbf{x}_{i_3}) + F \cdot (\mathbf{x}_{i_4} - \mathbf{x}_{i_5}) \quad (5)$$

Schemes (1) and (3), with notations as $DE/rand/1$ and $DE/current\ to\ best/2$, are the most often used in practice due to their good performance [2], [3].

B. Differential Evolution with Neighborhood Search (NSDE)

NSDE [1] is a recent DE variant that utilizes the neighborhood search (NS) strategy in evolutionary programming (EP). NS is a main strategy underpinning EP, and the characteristics of several NS operators' have been investigated in EP literature [11]. Although DE might be similar to the evolutionary process in EP, it lacks relevant concept of neighborhood search. NSDE is the same with the classical DE described in Section II.A except the scale factor F is replaced by the following equation:

$$F_i = \begin{cases} N_i(0.5, 0.5), & if\ U_i(0,1) < fp \\ \delta_i, & otherwise. \end{cases} \quad (6)$$

where i is the index of current trial vector, $U_i(0,1)$ stands for the uniform random number between 0 and 1, $N_i(0.5, 0.5)$ denotes a Gaussian random number with mean 0.5 and standard deviation 0.5, and δ_i denotes a Cauchy random variable with scale parameter $t = 1$. The parameter fp was set to a constant number 0.5 in NSDE.

The advantages of NS strategy in DE have been studied in [1]. Experimental results have shown that NSDE has significant advantages over classical DE on a broad range of different benchmark functions. It has been found that NSDE is effective in escaping from local optima when searching in environments without prior knowledge about what kind of search step size will be preferred.

C. Self-adaptive Differential Evolution (SaDE)

SaDE by Qin et al. [2], gives the first attempt to adopt two different mutation strategies in single DE variant. The motivation of SaDE is to solve the dilemma that mutation strategies involved in DE are often highly dependent on the problems under consideration. It introduces a probability p to control which mutation strategy to use, and p is gradually self-adapted according to the learning experience. Additionally, SaDE utilizes two methods to adapt and self-adapt DE's parameters F and CR. Detailed contributions of SaDE are summarized as follows:

1) Mutation strategies self-adaptation: SaDE selects mutation strategies Eq. (1) and Eq. (3) as candidates, and produces the trial vector based on:

$$\mathbf{v}_i = \begin{cases} Eq.\ (1), & if\ U_i(0,1) < p \\ Eq.\ (3), & otherwise. \end{cases} \quad (7)$$

Here p is set to 0.5 initially. After evaluation of all offspring, the number of offspring successfully entering the next generation while generated by Eq. (1) and Eq. (3) are recorded as ns_1 and ns_2, respectively, and the numbers of offspring discarded while generated by Eq. (1) and Eq. (3) are recorded as nf_1 and nf_2. Those two pairs of numbers are accumulated within a specified number of generations (50 in SaDE), called the "learning period". Then, the probability p is updated as:

$$p = \frac{ns_1 \cdot (ns_2 + nf_2)}{ns_2 \cdot (ns_1 + nf_1) + ns_1 \cdot (ns_2 + nf_2)} \quad (8)$$

Here ns_1, ns_2, nf_1 and nf_2 will be reset once p is updated after each learning period.

2) Scale factor F setting: In SaDE, F is set to

$$F_i = N_i(0.5, 0.3)$$

where $N_i(0.5, 0.3)$ denotes a Gaussian random number with mean 0.5 and standard deviation 0.3.

3) Crossover rate CR self-adaptation: SaDE allocates a CR_i for each individuals according to:

$$CR_i = N_i(CRm, 0.1) \quad (9)$$

CRm is set to 0.5 initially. These CR values for all individuals remain the same for several generations (5 in SaDE) and then a new set of CR values is generated using the same equation. During every generation, the CR values associated with offspring successfully entering the next generation are recorded in an array CR_{rec}. After a specified number of generations (25 in SaDE), CRm will be updated:

$$CRm = \frac{1}{|CR_{rec}|} \sum_{k=1}^{|CR_{rec}|} CR_{rec}(k) \quad (10)$$

CR_{rec} will be reset once CRm is updated. This self-adaptation scheme for CR is denoted as SaCR.

For detailed principles and explanations behind SaDE's self-adaptation strategies, parameter settings, or even simulated results, please refer to [2].

III. Self-adaptive Differential Evolution with Neighborhood Search

A. SaNSDE: The Incorporated Algorithms

It can be concluded that SaDE and NSDE have quite different emphases: The former pays special attention to self-adaptation between different mutation strategies, as well as the self-adaptation on crossover rate CR, while the latter intends to mix search biases of different NS operators through the parameter F, and no self-adaptation is adopted. The difference motivates us to introduce SaDE's self-adaptive mechanisms into NSDE, study their behaviors, and then propose a self-adaptive NSDE (SaNSDE).

Based on the motivations above, we address crucial issues of the proposed SaNSDE as follows:

1) Mutation strategies self-adaptation: SaNSDE utilizes the same method as SaDE in this part. For details, please refer to Eq. (7) and Eq. (8).
2) Scale factor F self-adaptation: SaNSDE inherits the method of controlling the parameter F from NSDE, but extending it to:

$$F_i = \begin{cases} N_i(0.5, 0.3), & if\ U_i(0,1) < fp \\ \delta_i, & otherwise. \end{cases}$$

where fp will be self-adapted as p is done in SaDE according to Eq. (8), except here we have to record corresponding F values that make offspring enter the next generation successfully.
3) Weighted crossover rate CR self-adaptation: We use a similar strategy to what SaDE does with SaCR strategy. But whenever we record a successful CR value in array CR_{rec}, we will also record the corresponding improvement on fitness value in array Δf_{rec}, with $\Delta f_{rec}(k) = f(k) - f_{new}(k)$. And then, Eq. (10) is changed to:

$$CRm = \sum_{k=1}^{|CR_{rec}|} w_k * CR_{rec}(k) \tag{11}$$

$$w_k = \Delta f_{rec}(k) / \left(\sum_{k=1}^{|\Delta f_{rec}|} \Delta f_{rec}(k) \right) \tag{12}$$

Note: here $|CR_{rec}| \equiv |\Delta f_{rec}|$. The weighted self-adaptation scheme for CR is denoted as SaCRW, and we will explain why we add the weight mechanism to the original SaCR in Section III.B with details.

Due to the significant successes of SaDE and NSDE, SaNSDE, which incorporates enhancements 1), 2) and 3), is promising.

B. Weighted CR Self-adaptation

The parameter CR of DE determines how many components of mutated vector will be introduced into current candidate for the next generation, so the probability of generating improved offspring from the same parent with a small CR is higher than that with a large CR. It can referred that the CRm of SaCR has an implicit bias towards small values during self-adaptation process. The bias might become harmful when optimizing nonseparable functions, in which interactions exist between variables. Because large CR value is required to change the nonseparable variables together.

To illustrate this problem, we conducted an experiment with *SaNSDE+SaCR* on the well-known Generalized Rosenbrock's function [11]. The evolution curves for "S runs" and "F runs" of 25 independent runs are given in Fig. 1. Here "S runs" means SaNSDE has found the region of optimum, while "F runs" means SaNSDE failed to do that. For "S runs", it can be found that CRm was successfully adapted to a large value, and after that the fitness values are improved quickly. For "F runs", CRm was adapted to a small value, and trapped there from that time on. The algorithm failed to make significant improvement on fitness values thereafter.

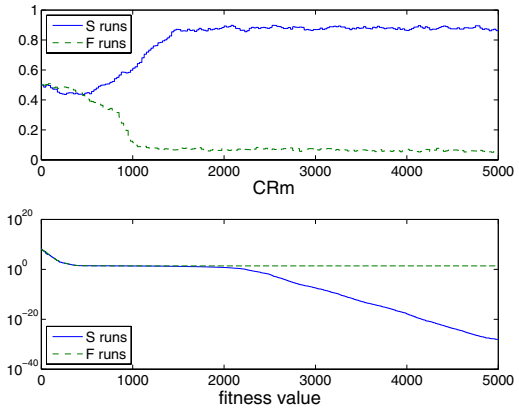

Fig. 1. Evolution curves of CRm and fitness value on the Generalized Rosenbrock function. "S runs" denotes results of successful runs, while "F runs" denotes results of failed runs of 25 independent runs. The vertical axes show the CRm value (up figure) and fitness value (down figure), while the horizontal axes show the number of generations.

On the other hand, it is assumed that large successful CR values will achieve larger improvement on fitness values than small successful CR values for nonseparable functions, because it will be good to change nonseparable variables together [1]. So we can balance the bias of SaCR with a weight based on the size of fitness value improvement. This is the basic motivation of SaCRW in Section III.A. To validate the effectiveness, another experiment with *SaNSDE+SaCRW* was conducted on the same function. The results are summarized in Table I. It can be found that SaCRW was successfully adapts CRm to required large values in all runs. The advantage is also shown by differences of fitness values.

TABLE I

SIMULATED RESULTS OF SACR AND SACRW ON THE GENERALIZED ROSENBROCK FUNCTION. THE RESULTS OF 25 INDEPENDENT RUNS ARE SORTED FROM 1ST TO 25TH BASED ON FITNESS VALUES

# of runs	SaNSDE+SaCR		SaNSDE+SaCRW	
	Fitness	Final CRm	Fitness	Final CRm
1st	7.42e-30	0.840	0.00e+00	0.888
5th	3.99e+00	0.051	0.00e+00	0.834
9th	2.33e+01	0.064	0.00e+00	0.853
13th	2.34e+01	0.057	0.00e+00	0.873
17th	2.37e+01	0.064	6.02e-30	0.964
21th	2.39e+01	0.053	1.24e-29	0.869
25th	2.43e+01	0.055	1.91e-29	0.879
Mean	1.82e+01	–	4.13e-30	–
Std	9.76e+00	–	6.28e-30	–

SaCRW made the algorithms success in all 25 runs, while SaCR made it achieve only 4 successful runs (1st–4th).

IV. EXPERIMENTAL STUDIES

A. Experimental Setup

Experimental validations for the proposed SaNSDE are conducted on both a set of classical test functions [11], and a new set of benchmark functions provided by CEC 2005 special session [12]. The algorithms used for comparison are SaDE, NSDE and SaNSDE (with SaCRW). The population size NP is set to 100 for all algorithms, and no other parameters is adjusted during evolution.

B. Results on Classical Benchmark Functions

The classical test set includes 23 functions, in which $f_1 - f_{13}$ are high-dimensional (30-D) and $f_{14} - f_{23}$ are low-dimensional functions. Functions $f_1 - f_5$ are unimodal, functions $f_8 - f_{13}$ are multimodal functions with many local optima, and functions $f_{14} - f_{23}$ are multimodal functions with only a few local optima. Details of these functions can be found in the appendix of [11]. The number of evolution generations of all algorithms is set to 1500 for $f_1 - f_4$, 5000 for f_5, 1500 for $f_6 - f_{13}$, 200 for f_{14}, 1500 for f_{15} and 200 for $f_{16} - f_{23}$. The average results of 25 independent runs are summarized in Tables II–IV.

TABLE II

EXPERIMENTAL COMPARISON ON f_1 – f_7 (OVER 25 RUNS).

Test Func	SaNSDE Mean	SaDE Mean	NSDE Mean	vs SaDE t-test	vs NSDE t-test
f_1	**3.02e-23**	7.49e-20	7.76e-16	$-2.23^{\dagger}$	$-9.25^{\dagger}$
f_2	**4.64e-11**	6.22e-11	4.51e-10	-0.94	$-13.39^{\dagger}$
f_3	**6.62e-22**	1.12e-18	1.06e-14	$-3.58^{\dagger}$	$-5.76^{\dagger}$
f_4	**1.59e-03**	2.96e-02	2.54e-02	$-20.21^{\dagger}$	$-11.97^{\dagger}$
f_5	**4.13e-30**	2.10e+01	1.24e+01	$-13.17^{\dagger}$	$-3.65^{\dagger}$
f_6	0.00e+00	0.00e+00	0.00e+00	0.00	0.00
f_7	**7.21e-03**	7.58e-03	1.20e-02	-0.53	$-7.14^{\dagger}$

† The value of t-test is significant at $\alpha = 0.05$ by a two-tailed test.

For unimodal functions $f_1 - f_7$, SaNSDE achieved much better results than SaDE and NSDE, except on the simple step function f_6, where all three algorithms performed exactly the same. The great difference can be seen from results on the Generalized Rosenbrock's Function, f_5. The evolution curves of parameter adaptation and fitness value for this function are given in Fig. 2 and 3. As we mentioned before, CRm in SaNSDE has been able to self-adapt to proper values.

TABLE III

EXPERIMENTAL COMPARISON ON f_8 – f_{13} (OVER 25 RUNS).

Test Func	SaNSDE Mean	SaDE Mean	NSDE Mean	vs SaDE t-test	vs NSDE t-test
f_8	-12569.5	-12569.5	-12569.5	0.00	0.00
f_9	1.84e-05	**4.00e-08**	7.97e-02	$4.31^{\dagger}$	-1.45
f_{10}	**2.36e-12**	9.06e-11	6.72e-09	$-6.24^{\dagger}$	$-18.24^{\dagger}$
f_{11}	**0.00e+00**	8.88e-18	4.68e-15	-1.00	$-2.69^{\dagger}$
f_{12}	**5.94e-23**	1.21e-19	5.63e-17	$-4.35^{\dagger}$	$-6.66^{\dagger}$
f_{13}	**3.12e-22**	1.75e-19	5.52e-16	$-5.39^{\dagger}$	$-4.91^{\dagger}$

† The value of t-test is significant at $\alpha = 0.05$ by a two-tailed test.

For multimodal functions $f_8 - f_{13}$, SaNSDE is the clear winner again, except that it was outperformed by SaDE on function f_9. With further observation of curves on Figs. 2 and 3, SaNSDE converged slower than SaDE, but still made good improvement all the way. This might have happened because the self-adapted parameters in SaNSDE need more time to find the proper values on this function.

TABLE IV

EXPERIMENTAL COMPARISON ON f_{14} – f_{23} (OVER 25 RUNS).

Test Func	SaNSDE Mean	SaDE Mean	NSDE Mean	vs SaDE t-test	vs NSDE t-test
f_{14}	0.998	0.998	0.998	0.00	0.00
f_{15}	3.07e-04	3.07e-04	3.07e-04	0.00	0.00
f_{16}	-1.03	-1.03	-1.03	0.00	0.00
f_{17}	0.398	0.398	0.398	0.00	0.00
f_{18}	3.00e+00	3.00e+00	3.00e+00	0.00	0.00
f_{19}	-3.39	-3.39	-3.39	0.00	0.00
f_{20}	-3.20	-3.20	-3.20	$-2.44^{\dagger}$	$2.50^{\dagger}$
f_{21}	-10.15	-10.15	-10.15	0.23	1.24
f_{22}	-10.40	-10.40	-10.40	0.99	1.05
f_{23}	-10.54	-10.54	-10.54	1.20	1.23

† The value of t-test is significant at $\alpha = 0.05$ by a two-tailed test.

Table IV shows the results for low-dimensional functions $f_{14} - f_{23}$. The three compared algorithms showed only very minor differences on f_{20} and $f_{21} - f_{23}$ (which cannot be seen from the mean values). That is because all of the algorithms have superior performance on this low-dimensional functions.

C. Results on CEC 2005 Benchmark Functions

To evaluate SaNSDE further, a new set of benchmark functions provided by CEC 2005 special session was used. It includes 25 functions with different complexity [12]. Functions $f_{cec1} - f_{cec5}$ are unimodal while the remaining 20 functions are multimodal. Since functions $f_{cec15} - f_{cec25}$ are hybrid composition functions, which are very time consuming for fitness evaluation, we only used the first 14 functions of the set in our experiments. All of these functions are scalable, and we set their dimensions to 30 in our experiments. Detailed description of these functions can be found in [12]. The number of evolution generations is set to 3000 for all

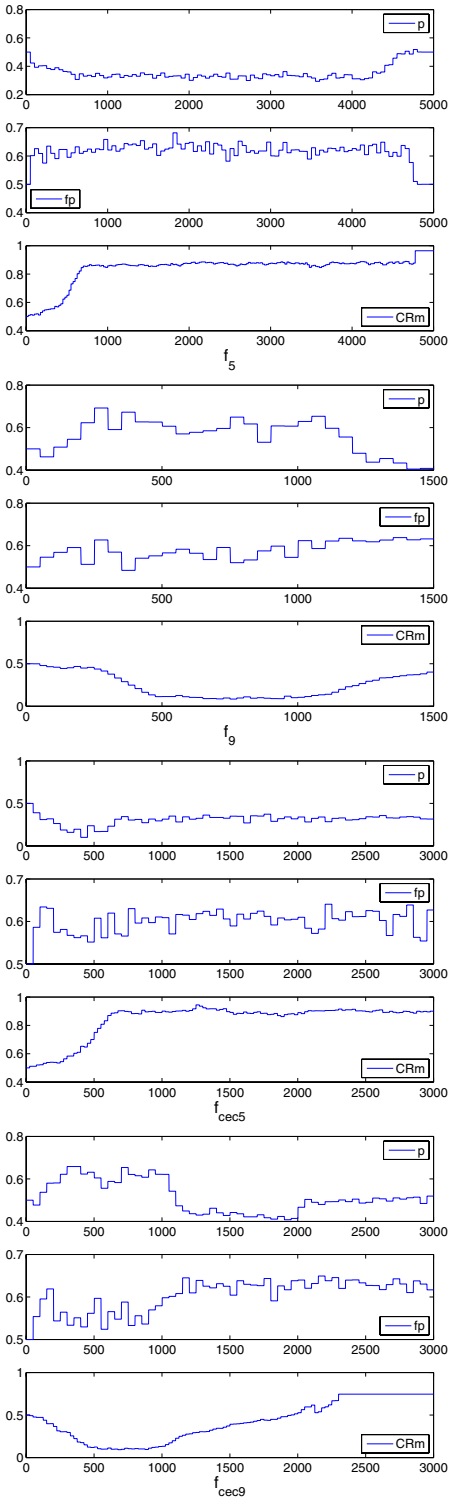

Fig. 2. The self-adaptation curves of p, fp and CRm for f_5, f_9, f_{cec5}, and f_{cec9}. On the vertical axes are shown their values (between 0 and 1), while on the horizontal axes are shown the number of generations.

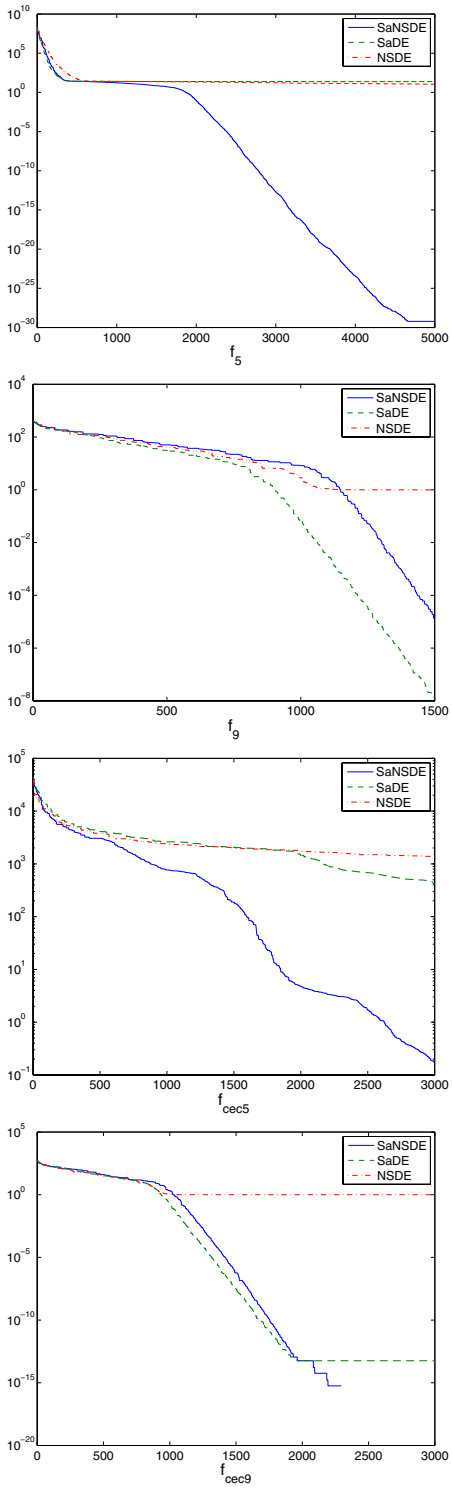

Fig. 3. The evolution curves for f_5, f_9, f_{cec5} and f_{cec9}. The vertical axes show the distance to the optimum and the horizontal axes show the number of generations.

functions. Error value, i.e. the difference between current fitness value and optimum, is used to compare algorithm's performance. The average error values of 25 independent runs are summarized in Tables V and VI.

TABLE V
EXPERIMENTAL COMPARISON ON $f_{cec1} - f_{cec5}$ (OVER 25 RUNS).

Test Func	SaNSDE Error	SaDE Error	NSDE Error	vs SaDE t-test	vs NSDE t-test
f_{cec1}	0.00e+00	0.00e+00	0.00e+00	0.00	0.00
f_{cec2}	**5.68e-14**	1.25e-13	4.11e+00	$-7.86^{\dagger}$	$-8.23^{\dagger}$
f_{cec3}	**5.43e+04**	1.77e+05	1.67e+06	-0.86	$-11.54^{\dagger}$
f_{cec4}	**1.22e-04**	1.89e+02	8.27e+01	-1.00	$-7.06^{\dagger}$
f_{cec5}	**2.45e-01**	1.00e+03	1.15e+03	$-3.80^{\dagger}$	$-11.61^{\dagger}$

† The value of t-test is significant at $\alpha = 0.05$ by a two-tailed test.

For unimodal functions, SaNSDE performed better than the other two algorithms on all 5 functions, and significantly better on functions f_{cec2}, f_{cec4} and f_{cec5}. This is consistent with conclusions drawn on classical unimodal functions. The effectiveness and efficiency on f_{cec5} can be seen from evolution curves in Fig. 2 and 3.

TABLE VI
EXPERIMENTAL COMPARISON ON $f_{cec6} - f_{cec14}$ (OVER 25 RUNS).

Test Func	SaNSDE Error	SaDE Error	NSDE Error	vs SaDE t-test	vs NSDE t-test
f_{cec6}	**1.59e-01**	2.99e+01	2.89e+01	$-8.13^{\dagger}$	$-5.86^{\dagger}$
f_{cec7}	**8.57e-03**	1.65e-02	1.12e-02	-1.85	-0.90
f_{cec8}	2.09e+01	2.09e+01	2.09e+01	-0.19	0.08
f_{cec9}	**0.00e+00**	2.27e-15	1.99e-01	-1.00	$-2.45^{\dagger}$
f_{cec10}	**4.21e+01**	5.15e+01	4.24e+01	$-2.69^{\dagger}$	-0.10
f_{cec11}	**1.02e+01**	2.71e+01	1.48e+01	$-29.81^{\dagger}$	$-3.43^{\dagger}$
f_{cec12}	**4.06e+04**	4.28e+04	1.74e+05	-0.99	$-23.90^{\dagger}$
f_{cec13}	2.12e+00	**2.00e+00**	5.11e+00	$2.88^{\dagger}$	$-31.55^{\dagger}$
f_{cec14}	1.27e+01	**1.26e+01**	1.32e+01	1.35	$-10.56^{\dagger}$

† The value of t-test is significant at $\alpha = 0.05$ by a two-tailed test.

For multimodal functions, SaNSDE obtained better results on almost all functions, except on f_{cec13} and f_{cec14}, where it was outperformed by SaDE. All algorithms performed badly on the two functions, and SaDE happened to show a minor superiority. f_{cec13} and f_{cec14} are expanded functions, which are composed of other different functions. This makes their characteristics unclear for further case study. The analysis of algorithms' evolutionary behaviors on functions like them is one of the focuses of our future work. Fig. 3 shows the evolution curves of f_{cec9}. It can be seen that SaNSDE found the optimum in less than the maximum number of available generations. The curves in Fig. 2 of this functions showed SaNSDE required different values for p, fp and CRm during different stages, and the self-adaptation strategies were able to adjust these parameters as needed (from large to small, then to large again).

V. CONCLUSIONS AND DISCUSSIONS

In this paper, we proposed a new self-adaptive DE variant, SaNSDE, which is an improved version of our previous algorithm NSDE. The SaNSDE can be viewed as a hybridization of SaDE [2] and NSDE [1]. In SaNSDE: 1) We utilized the self-adaptation strategy of SaDE to adapt between candidate mutations; 2) We applied a self-adaptation to adjust parameter F; 3) We illustrated the ill-condition of original CR self-adaptation in SaDE, and proposed an enhanced version with weighting. The performance of the proposed SaNSDE algorithm is evaluated and discussed on both a set of 23 classical test functions[11], and a new set of 14 benchmark functions provided by CEC 2005 special session [12]. SaNSDE has shown significant superiority over both SaDE and NSDE.

Besides the SaDE mentioned in this paper, several other self-adaptive DE variants (SaDEs) have also been proposed. Omran et al. proposed a SDE [13], [14] by adapting parameters F and CR based on normal distribution. Brest et al. presented the jDE, which attaches F and CR values to all individuals of population, and evolves these control parameters at individual level [9]. F and CR are updated in each generation according to some heuristic rules. In their later work [15], an improved version of jDE, namely jDE-2, has also been proposed by importing the mutation strategies self-adaptation from SaDE (Qin et al.). In some respects, SaNSDE is the inheritor of NSDE and SaDE, while it is different from other self-adaptive DE algorithms in two major aspects: 1) By mixing the search biases of both Gaussian and Cauchy operators, SaNSDE considers a trade-off between small and large search step sizes; 2) SaNSDE self-adapts all its control parameters according to statistical learning experience during evolution, rather than other heuristic updating rules. We have compared the performance of SaNSDE with these latest SaDEs, but the lack of space prevents showing the results of those experiments. In general, SaNSDE achieved comparable results to that of the other methods.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (Grant No. 60428202), the Fund for Foreign Scholars in University Research and Teaching Programs (Grant No. B07033), and the Graduate Innovation Fund of University of Science and Technology of China (Grant No. KD2007044).

REFERENCES

[1] Z. Yang, J. He, and X. Yao, "Making a Difference to Differential Evolution," in *Advances in Metaheuristics for Hard Optimization*, Z. Michalewicz and P. Siarry, Eds. Springer, 2008, pp. 397–414.

[2] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1785–1791, 2005.

[3] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, ISBN:3-540-20950-6, 2005.

[4] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," *Proceedings of the 2004 Congress on Evolutionary Computation*, vol. 2, pp. 1980–1987, 2004.

[5] N. Hansen, "Compilation of results on the CEC benchmark function set," *Institute of Computational Science, ETH Zurich, Switerland, Tech. Rep*, vol. 13, 2005.

[6] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 1, pp. 22–34, 1999.

[7] D. Zaharie, "Critical Values for the Control Parameters of Differential Evolution Algorithms," *Proceedings of the 8th International Conference on Soft Computing*, pp. 62–67, 2002.
[8] R. Gamperle, S. Muller, and P. Koumoutsakos, "A Parameter Study for Differential Evolution," *Proceedings WSEAS international conference on advances in intelligent systems, fuzzy systems, evolutionary computation*, pp. 293–298, 2002.
[9] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Žumer, "Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems," *IEEE Transactions on Evolutionary Computation*, vol. 2, pp. 82–102, 2006.
[10] T. Bäck and H. P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.
[11] X. Yao, Y. Liu, and G. Lin, "Evolutionary Programming Made Faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.
[12] 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 CEC 2005 Special Session on Real-Parameter Optimization," *Technical Report, Nanyang Technological University, Singapore, http://www.ntu.edu.sg/home/EPNSugan*, 2005.
[13] M. Omran, A. Salman, and A. Engelbrecht, "Self-adaptive Differential Evolution," *Proceedings of the 2005 International Conference on Computational Intelligence and Security*, pp. 192–199, 2005.
[14] A. Salman, A. Engelbrecht, and M. Omran, "Empirical analysis of self-adaptive differential evolution," *European Journal of Operational Research*, vol. 183, no. 2, pp. 785–804, 2007.
[15] J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. Maučec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 11, no. 7, pp. 617–629, 2007.