Making a Difference to Differential Evolution

Zhenyu Yang[1], Jingsong He[1] and Xin Yao[1,2]

[1] Nature Inspired Computation and Applications Laboratory (NICAL), University of Science and Technology of China, Hefei, Anhui, China. zhyuyang@mail.ustc.edu.cn, hjss@ustc.edu.cn

[2] School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK. x.yao@cs.bham.ac.uk

Abstract

Differential evolution (DE) and evolutionary programming (EP) are two major algorithms in evolutionary computation. They have been applied with success to many real-world numerical optimization problems. Neighborhood search (NS) is a main strategy underpinning EP. There have been analyses of different NS operators' characteristics. Although DE might be similar to the evolutionary process in EP, it lacks the relevant concept of neighborhood search. In this chapter, DE with neighborhood search (NSDE) is proposed based on the generalization of NS strategy. The advantages of NS strategy in DE are analyzed theoretically. These analyses mainly focus on the change of search step size and population diversity after using neighborhood search. Experimental results have shown that DE with neighborhood search has significant advantages over other existing algorithms on a broad range of different benchmark functions. NSDE's scalability is also evaluated on a number of benchmark problems, whose dimension ranges from 50 to 200.

Key words: Differential Evolution, Global Optimization, Evolutionary Algorithms, Neighbourhood Search, Hybrid Algorithms

1 Introduction

Differential evolution (DE) is a simple yet effective global optimization algorithm with superior performance in several real-world applications [1,2]. Several variations of DE have been proposed to improve its performance. A self-adaptive strategy has also been investigated to adapt between different variants of DE [3]. Although there are only three strategy parameters (population size NP, crossover rate CR and mutation scaling factor F), it was found that the performance of DE is very sensitive to the setting of these control parameters. Zaharie [4] analyzed how these control parameters influence the population diversity of DE, while [5] used extensive experiments

to study how the performance of DEs is affected by these factors. Although empirical rules can be found for choosing these control parameters, they are not general and therefore not suitable for practical applications.

There are two main steps in DE: mutation and crossover. The mutation will create a trial vector for each individual, and then crossover will recombine each pair of trial vectors and individuals to produce offspring. Since DE only uses discrete recombination for crossover, mutation supplies the major power to make progress during evolution. As another mutation-based strategy, evolutionary programming (EP) is also a major branch of evolutionary computation. In EP, new offspring are obtained by giving a perturbation to the original individuals. That means all offspring for the next generation are generated in the neighborhood of current solutions. Thus EP uses a neighborhood search (NS) strategy to improve the quality of solutions. The characteristics of different NS operators have been analyzed in [6,7]. EP's evolutionary behavior has shown that NS is an efficient operator for such a generate-and-test method. However there is no definite neighborhood search concept in DE, and little work has been done to investigate how this NS strategy will affect DE's evolutionary behaviors. Based on the success of the NS strategy in EP, the idea of DE with a similar strategy deserves more investigation.

In this chapter, the common features of DE and EP are generalized into a uniform framework, and based on this understanding, DE with neighborhood search (NSDE) is proposed to improve its neighborhood search ability. Then the advantages of DE with neighborhood search are analyzed theoretically. These analyses mainly focus on the change of search step size and population diversity after using a neighborhood search (NS) strategy. Experimental evidence is also given to regarding the evolutionary behavior of DE with neighborhood search on widely used benchmark functions.

2 Preliminaries

2.1 Evolutionary Programming

Optimization by evolutionary programming (EP) can be summarized into two major steps: first mutate the solutions in the current population, and then select the next generation from the mutated and the current solutions [8].

$$\boldsymbol{x}_i'(j) = \boldsymbol{x}_i(j) + \boldsymbol{\eta}_i(j)N_j(0,1) \tag{1}$$

$$\boldsymbol{\eta}_i'(j) = \boldsymbol{\eta}_i(j)\exp(\tau' N(0,1) + \tau N_j(0,1)) \tag{2}$$

where $i \in \{1,\ldots,\mu\}$, μ is the population size, $j \in \{1,\ldots,n\}$, n is the dimension of object parameters, $N(0,1)$ denotes a normally distributed one-dimensional random number with mean zero and standard deviation one, and $N_j(0,1)$ indicates that the random number is generated anew for each value of j. Usually, evolutionary programming using Eqs. (1) and (2) is called classical evolutionary programming (CEP) .

The Cauchy operator is introduced into Eq. (1) to substitute Gaussian mutation in [6], i.e. the update equation is replaced by:

$$\boldsymbol{x}_i'(j) = \boldsymbol{x}_i(j) + \boldsymbol{\eta}_i(j)\delta_j \tag{3}$$

where δ_j is a Cauchy random variable with the scale parameter $t = 1$ and is generated anew for each j. EP using Cauchy mutations is called fast evolutionary programming (FEP). IFEP (Improved FEP) based on mixing different mutation operators, and LEP based on mutations with the Lévy probability distribution were also proposed in [6] and [7], respectively.

By Eq. (1) or Eq. (3), we can find that new offspring are obtained by giving a perturbation to the original individual. This means all offspring for the next generation are generated in the neighborhood of current solutions. Thus EP improves the quality of solutions through a neighborhood search (NS) strategy. Operators such as Gaussian $N_j(0,1)$ and Cauchy δ_j are used to control the size and shape of the neighborhood: η_j is a self-adaptive scaling factor for the neighborhood size. The characteristics of different NS operators have been analyzed in [6,7]. How these NS operators are used will strongly affect EP's performance. Based on the importance of the NS strategy in EP, we intend to investigate whether such an NS strategy can be generalized and used in other evolutionary algorithms.

2.2 Differential Evolution

Individuals are represented by D-dimensional vectors $\boldsymbol{x}_i, \forall i \in \{1, \ldots, NP\}$ in DE, where D is the number of optimization parameters and NP is the population size. According to the description by Storn and Price [1], the mutation and crossover of original DE can be summarized as follows:

$$\boldsymbol{y}_i = \boldsymbol{x}_{i_1} + (\boldsymbol{x}_{i_2} - \boldsymbol{x}_{i_3}) \cdot F \tag{4}$$

$$\boldsymbol{z}_i(j) = \begin{cases} \boldsymbol{y}_i(j), & \text{if} \quad U_j(0,1) < CR \\ \boldsymbol{x}_i(j), & \text{otherwise} \end{cases} \tag{5}$$

with $i, i_1, i_2, i_3 \in [1, NP]$ are integers and mutually different. $F > 0$ is a real constant factor to control the differential variation $\boldsymbol{d}_i = \boldsymbol{x}_{i_2} - \boldsymbol{x}_{i_3}$, and $U_j(0,1)$ denotes a uniform random number between 0 and 1. To represent crossover result $\boldsymbol{z}_i$ more formally, we can define a Boolean mask $\mathbf{M} = (\mathbf{M}(1), \mathbf{M}(2), \ldots, \mathbf{M}(D))$ as follows:

$$\mathbf{M}(j) = \begin{cases} 1, & \text{if} \quad U_j(0,1) < CR \\ 0, & \text{otherwise.} \end{cases}$$

and then $\boldsymbol{z}_i$ can be represented as:

$$\begin{aligned} \boldsymbol{z}_i &= \boldsymbol{y}_i \cdot \mathbf{M} + \boldsymbol{x}_i \cdot \overline{\mathbf{M}} &&= \boldsymbol{y}_i \cdot \mathbf{M} + \boldsymbol{x}_i \cdot (\mathbf{I} - \mathbf{M}) \\ &= \boldsymbol{y}_i \cdot \mathbf{M} + \boldsymbol{x}_i - \boldsymbol{x}_i \cdot \mathbf{M} &&= \boldsymbol{x}_i - (\boldsymbol{x}_i - \boldsymbol{y}_i) \cdot \mathbf{M} \end{aligned}$$

where $\mathbf{I}$ is an all-'1' vector $\mathbf{I} = (1, \ldots, 1)_D$. With these equations, we know that $\mathbf{M}$ is a componential mask of vector $(\boldsymbol{x}_i - \boldsymbol{y}_i)$. For each offspring, $\boldsymbol{z}_i(j) = \boldsymbol{x}_i(j)$ if $\mathbf{M}(j) = 0$, otherwise $\boldsymbol{z}_i(j) = \boldsymbol{y}_i(j)$. This means crossover in DE can be regarded as a selection process on the mutated components. After mutation DE performs a subspace selection process through crossover, and then it will search for better solutions in the subspace.

Now, similarity can be found between the evolutionary processes of DE and EP. Although there is no concept of neighborhood in DE, it has been carried out with a scaling factor F that has some relation with DE's search step size [5]. A large F value is expected to increase the probability of escaping from local optima. However, it also increases the perturbation of mutation, which will decrease DE's convergence speed. This is similar to the characteristics of neighborhood search operators used in EP, except there F was restricted to a constant number. So within the selected subspace after crossover, DE will have similar evolutionary behaviour to EP. To make use of the successful neighborhood search (NS) operators in EP, the idea of DE with similar NS strategy deserves more investigation.

3 DE with Neighborhood Search

3.1 Analyses of DE's Search Characteristics

In Eq. (4), note that the smaller the difference between parameters $\boldsymbol{x}_{i_2}$ and $\boldsymbol{x}_{i_3}$, the smaller the difference vector $\boldsymbol{d}_i = \boldsymbol{x}_{i_2} - \boldsymbol{x}_{i_3}$ and therefore the perturbation. That means if the population becomes close to the optimum, the step length is automatically decreased. This is similar to the self-adaptive step size control found in evolutionary programming. Based on this understanding, we can use an uniform mutation equation for DE and EP as follows:

$$\boldsymbol{x}_i'(j) = \boldsymbol{x}_i(j) + \xi_i(j)\ \psi \tag{6}$$

where $\xi_i(j)$ means $\boldsymbol{d}_i(j)$ ($\boldsymbol{d}_i$ is the difference vector $\boldsymbol{x}_{i_2} - \boldsymbol{x}_{i_3}$), and is $\boldsymbol{\eta}_i(j)$ in EP. ψ is a constant number F in DE, a Gaussian random number $N_j(0, 1)$ in CEP , and a Cauchy random number δ_j in FEP .

Generalizing the analysis method for the mean search step size in [6], the expected length of ψ jumps in the universal equation can be calculated as follows:

$$E_\psi = \int_{-\infty}^{+\infty} x\ \Psi(x)\ \mathrm{d}x$$

where $\Psi(x)$ is the distribution density function for generating the number ψ. When $\Psi(x)$ takes a Gaussian function and a Cauchy function , the expected length of Gaussian and Cauchy jumps are 0.8 and $+\infty$, respectively. Obviously, $\Psi(x)$ takes an impulse function $\delta(x - F)$ in DE and the expected jump will be:

$$E_{DE} = \int_{-\infty}^{+\infty} x\ \delta(x - F)\ \mathrm{d}x = F$$

Up to now we have shown the relation between scaling factor F and search step size theoretically. After setting a value for F, the search step size will be determined directly. Since different optimization problems or even different stages of the same evolution may demand different kinds of search step size, it is easy to understand why the empirical value $F = 0.5$ is not always suitable, and DE with more universal NS operators will have greater potential.

3.2 Mixing Search Biases of Different NS Operators

In Sect. 3.1, we have given a uniform equation for DE and EP, and within the frame we can generalize Eq. (4) to:

$$\boldsymbol{y}_i = \boldsymbol{x}_{i_1} + \boldsymbol{d}_i \cdot \psi \tag{7}$$

where ψ denotes a NS operator, which can be substituted with any kind of neighborhood search operator. The NS operator is used to control the size and shape of neighborhood during the evolutionary process. Assume the density function of ψ is $f_\psi(x)$, then the probability of generating jumps l smaller than a specified step L and larger than L will be:

$$P(l < L) = \int_{-L}^{+L} f_\psi(x)\ \mathrm{d}x$$
$$P(l > L) = \int_{-\infty}^{-L} f_\psi(x)\ \mathrm{d}x + \int_{+L}^{+\infty} f_\psi(x)\ \mathrm{d}x$$

With these equations, the probability of generating different jumps by the NS operator can be calculated, and thus we can achieve some basic search biases for different NS operators. To show the expected advantages of changing the constant number F to a NS operator, two widely used NS operator candidates will be analyzed here. They are Gaussian random $N(0,1)$, and Cauchy random $C(t=1)$. The probabilities of them generating different jumps are given in Fig. 1.

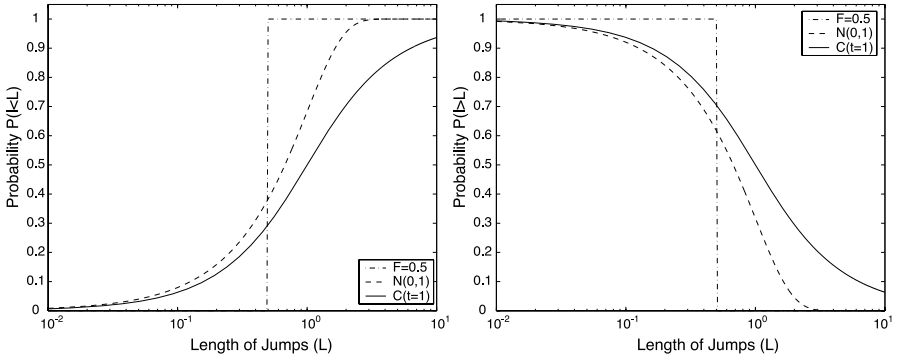


Fig. 1 The probability of generating different jumps for the given NS operators.

First, from Fig. 1 we can observe that all of the NS operators are more flexible than the constant setting for F. Constant F can only produce fixed length jump steps, while the two NS operators have the ability to produce many kinds of jump steps with different probabilities. For a common optimization problem, the probability is very low that the fixed jump steps produced by F are just right for evolution. So NS operators will be more universal than a constant setting for F. Deeper observation shows that Gaussian random $N(0,1)$ is very localized. It is more likely to produce small jumps. In contrast, Cauchy random $C(t=1)$ is more expandable. It has a greater probability of producing long jumps. Small jumps will be beneficial when the current search point is near the global optimum. However, as analyzed in [6], convergence of the evolution process will be very slow and will risk being trapped in some of the local optima. Long jumps have the ability to escape from poor local optima and locate a good near-global optimum. But this will be beneficial only when the global is sufficiently far away from the current search point. Generally, the Cauchy operator will perform better when far away from the global optimum, while the Gaussian operator is better at finding a local optimum in a good region. It will therefore be beneficial to mix different search methods but to also bias them differently. Here we change DE's mutation to:

$$\boldsymbol{y}_i = \boldsymbol{x}_{i_1} + \begin{cases} \boldsymbol{d}_i \cdot N(0.5, 0.5), & \text{if} \quad U(0,1) < 0.5 \\ \boldsymbol{d}_i \cdot \delta, & \text{otherwise.} \end{cases} \tag{8}$$

where $N(0.5, 0.5)$ denotes a normally distributed one-dimensional random number with mean 0.5 and standard deviation 0.5, and δ is a Cauchy random variable with scale parameter $t = 1$. The parameters $(0.5, 0.5)$ for the Gaussian operator are taken after determining an empirical value for F in DE. Equation (8) introduces a neighborhood search strategy that can produce many more kinds of step sizes, and this is expected to be beneficial when dealing with real-world optimization problems. Here the neighborhood search inspired DE is denoted NSDE .

Population diversity can be used to analyze the expected advantages of DE with neighborhood search. It has been found that if $\psi \sim F \cdot N(0,1)$, the expected population variance after mutation and crossover becomes:

$$E(\mathrm{Var}(Y)) = \left(2F^2 + \frac{m-1}{m}\right)\mathrm{Var}(x) \tag{9}$$

$$E(\mathrm{Var}(Z)) = \left(2F^2 p - \frac{2p}{m} + \frac{p^2}{m} + 1\right)\mathrm{Var}(x) \tag{10}$$

where m is the dimension of object parameters, and p is the value of crossover rate CR. These equations are proved in [4] in detail.

Similar analysis can be carried out when ψ is restricted to a constant number F. For $E((F \cdot N(0,1))^2) = E(F^2) = F^2$, the result for $E(Var(Y))$ and $E(Var(Z))$ will remain the same as Eqs. (9) and (10), i.e. the population diversity stays the same after using the Gaussian NS operator. In Eq. (4), the mutation will be determined by d_i directly. For each mutation, the object parameters can only change with step d_i. After

introducing the Gaussian NS operator, DE's search step size is determined not only by difference vector d_i, but also by the NS operator $N(0,1)$. The operator can scale d_i into many kinds of search step sizes, while the original constant number can only produce a fixed step size $\boldsymbol{d}_i \cdot F$. This will be an advantage when DE is searching for the global optimum in an unknown environment, where no prior knowledge exists about what search step size is prefered.

However, a localized Gaussian operator will only be beneficial when near the small neighborhood of the global optimum. In contrast, the Cauchy operator can overcome this limitation. For the Cauchy variable with scale parameter $t = 1$, we know that

$$\mathrm{Var}(\delta) = \int_{-\infty}^{+\infty} x^2 \frac{1}{\pi(1+x^2)}\, \mathrm{d}x = +\infty$$

$$\mathrm{E}(\delta^2) = Var(\delta) + (\mathrm{E}(\delta))^2 = +\infty$$

So after changing ψ in Eq. (7) to a Cauchy random variable, it is easy to carry out the expected population variance after mutation, and crossover that becomes:

$$\mathrm{E}(\mathrm{Var}(Y)) = E(\mathrm{Var}(Z)) = +\infty$$

Obviously, the Cauchy operator is much more global than the Gaussian, so DE with a Cauchy NS operator will have superior ability to escape from local optima. Equation (8) has considered not only a Gaussian operator , but also the Cauchy operator's search biases, so NSDE is expected to be more powerful when searching in an environment without prior knowledge. In the next section, experiments will be provided to test the performance of NSDE on a set of widely used benchmark functions.

4 Experimental Studies

Here experimental evidence is provided to study how neighborhood search operators influence DE's performance. We evaluate the performance of the proposed NSDE algorithm on both classical test functions and the new set of functions provided by CEC2005 special session. NSDE follows Eq. (8) to replace F with NS operators, and CR is set to U(0,1) to enhance DE's subspace search ability, here U(0,1) stands for a uniform random between 0 and 1. The algorithms used for comparison are classical DE with empirical parameter setting [5, 9] and other widely used algorithms such as FEP, CMAES .

4.1 NSDE on classical benchmark functions

First we test NSDE's performance on a set of 23 classical functions for numeric optimization. Functions $f_1 - f_{13}$ are high-dimensional problems. Functions $f_1 - f_5$ are unimodal. Function f_6 is the step function which has one minimum and is discontinuous. Function f_7 is a noisy quartic function where $random[0,1)$ is a uniformly

distributed random variable in $[0, 1)$. Functions $f_8 - f_{13}$ are multimodal functions where the number of local minima increases exponentially with the problem dimension [6,7]. Functions $f_{14} - f_{23}$ are low-dimensional functions which have only a few local minima [6,7]. Details of these functions can be found in the appendix to [6]. The average experimental results of 50 independent runs are summarized in Tables 1–3 (the results for FEP are taken from [6]).

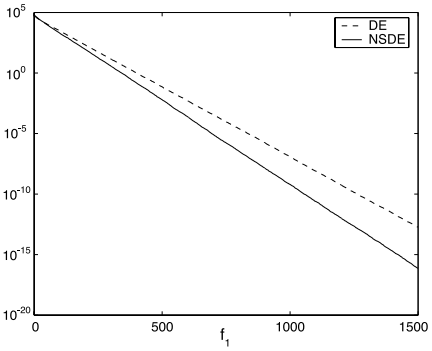
For unimodal functions $f_1 - f_7$, it is apparent that the proposed NSDE performs better than both DE and FEP. In particular, NSDE's results on $f_1 - f_4$ are significantly better than DE and FEP. It seems a large neighborhood search is efficient in speeding up the evolutionary convergence on unimodal functions. However, NSDE performs worse than DE on f_5. This is a generalized Rosenbrock's function, and there are correlations between each pair of neighboring object parameters. It can be inferred that the one-dimension based neighborhood search strategy has difficulty in optimizing such functions. No strong conclusion can be drawn for f_6 and f_7. There is no significant difference among all the algorithms.

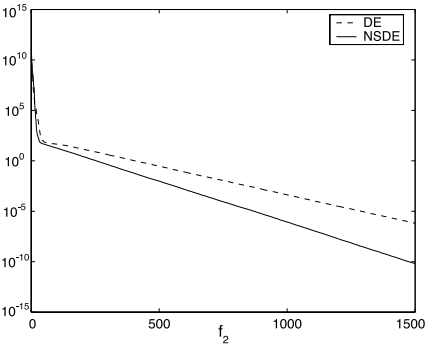
Table 2 shows the experimental results for functions $f_8 - f_{13}$. These functions are multimodal functions with many local optima. Their landscapes appear to be very "rugged" [6], and are often regarded as being difficult to optimize. Figure 3 shows the evolutionary processes of NSDE and DE for these functions. It can be observed that DE stagnates rather early in the search and makes little progress thereafter, while NSDE keeps making improvements throughout the evolution. It appears that DE is trapped in one of the local optima and is unable to get out. NSDE on the other hand, has the ability to produce many kinds of search step sizes with neighborhood search operators and thus has a higher probability of escaping from a local optimum when trapped. A good near-global optimum is more likely to be found by NSDE. In terms of detailed results in Table 2, NSDE performs significantly better than DE and FEP on almost all these functions. It can be concluded that DE with neighborhood search is more effective and efficient when searching in mutimodal functions with many optima.

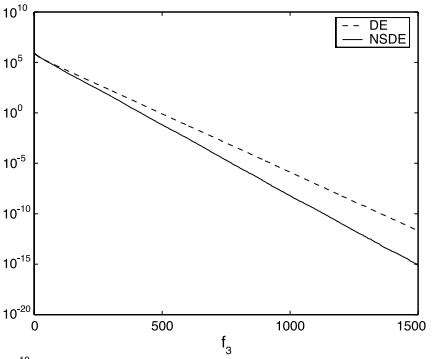
For multimodal functions with only a few local optima, such as $f_{14} - f_{23}$, both NSDE and DE have better performance than FEP, except that the three algorithms performed exactly the same on f_{16}, f_{17} and f_{19}. NSDE have a similar performance to classical DE. They performed exactly the same on six (i.e. $f_{14} - f_{19}$) out of ten functions. For the rest of the functions, NSDE performed better on f_{20}, but was outperformed by DE on $f_{21} - f_{23}$. To trace why NSDE is inefficient on these 3 functions, further experiments are conducted to observe its evolutionary behaviors. In these experiments we give more computational effort to NSDE, and the results of 50 independent runs are summarized in Table 4.

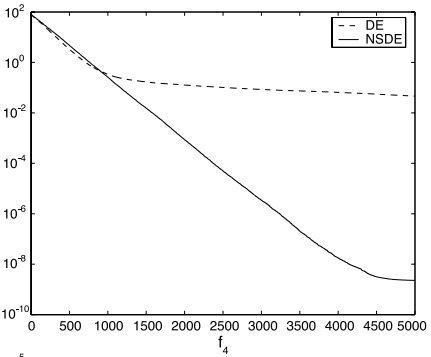
With a few more generations, NSDE can find the global optima of these functions. Figure 4 shows the evolutionary processes for $f_{21} - f_{23}$. It is clear that NSDE's

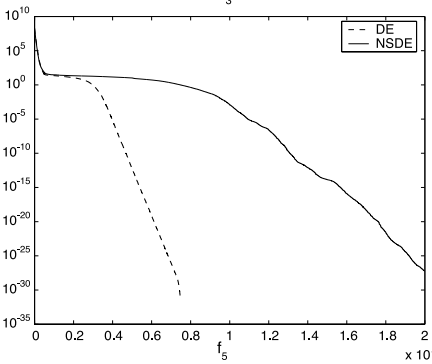
Fig. 2 Evolution process of the mean best values found for unimodal functions $f_1 - f_7$. The results were averaged over 50 runs. The *vertical axis* is the function value and the *horizontal axis* is the number of generations

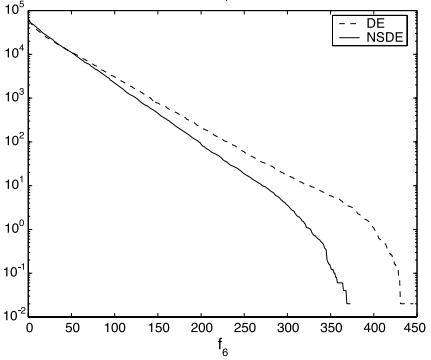

DE
NSDE
10^{5}
10^{0}
10^{-5}
10^{-10}
10^{-15}
10^{-20}
0
500
1000
1500
f_1


DE
NSDE
10^{15}
10^{10}
10^{5}
10^{0}
10^{-5}
10^{-10}
10^{-15}
0
500
1000
1500
f_2


DE
NSDE
10^{10}
10^{5}
10^{0}
10^{-5}
10^{-10}
10^{-15}
10^{-20}
0
500
1000
1500
f_3


DE
NSDE
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
f_4


DE
NSDE
10^{10}
10^{5}
10^{0}
10^{-5}
10^{-10}
10^{-15}
10^{-20}
10^{-25}
10^{-30}
10^{-35}
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
x 10^{4}
f_5


DE
NSDE
10^{5}
10^{4}
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
0
50
100
150
200
250
300
350
400
450
f_6

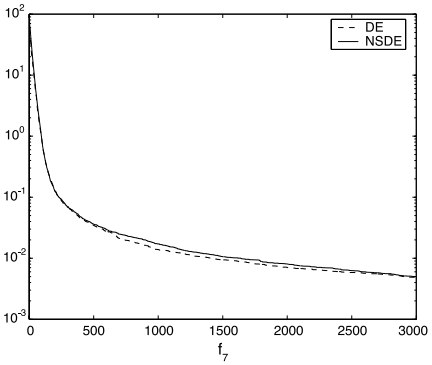

DE
NSDE
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
0
500
1000
1500
2000
2500
3000
f_7

Table 1 Comparison between NSDE, DE and FEP on $f_1 - f_7$. All results have been averaged over 50 independent runs

Test Func	# of Gen's	NSDE Mean	DE Mean	# of Gen's	FEP Mean	vs DE t-test	vs FEP t-test
f_1	1500	7.10×10^{-17}	1.81×10^{-13}	1500	5.70×10^{-04}	$-10.84^{\dagger}$	$-31.00^{\dagger}$
f_2	1500	6.49×10^{-11}	6.43×10^{-07}	2000	8.10×10^{-03}	$-17.22^{\dagger}$	$-74.38^{\dagger}$
f_3	1500	7.86×10^{-16}	2.12×10^{-12}	5000	1.60×10^{-02}	$-10.86^{\dagger}$	$-8.08^{\dagger}$
f_4	5000	2.27×10^{-09}	4.61×10^{-02}	5000	0.3	$-2.05^{\dagger}$	$-4.24^{\dagger}$
f_5	20,000	5.90×10^{-28}	0	20,000	5.06	1.03	$-6.10^{\dagger}$
f_6	1500	0	0	1500	0	0	0
f_7	3000	4.97×10^{-03}	4.84×10^{-03}	3000	7.60×10^{-03}	0.52	$-6.48^{\dagger}$

† The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test

Table 2 Comparison between NSDE, DE and FEP on $f_8 - f_{13}$. All results have been averaged over 50 independent runs

Test Func	# of Gen's	NSDE Mean	DE Mean	# of Gen's	FEP Mean	vs DE t-test	vs FEP t-test
f_8	1500	− 12,569.5	− 11,362.1	9000	− 12,554.5	$-5.08^{\dagger}$	$-2.02^{\dagger}$
f_9	3000	3.98×10^{-02}	138.70	5000	4.60×10^{-02}	$-39.63^{\dagger}$	−0.22
f_10	1500	1.69×10^{-09}	1.20×10^{-07}	1500	1.80×10^{-02}	$-19.86^{\dagger}$	$-60.61^{\dagger}$
f_11	1500	5.80×10^{-16}	1.97×10^{-04}	2000	1.60×10^{-02}	−1.00	$-5.14^{\dagger}$
f_12	1500	5.37×10^{-18}	1.98×10^{-14}	1500	9.20×10^{-06}	$-6.79^{\dagger}$	$-18.07^{\dagger}$
f_13	1500	6.37×10^{-17}	1.16×10^{-13}	1500	1.60×10^{-04}	$-7.39^{\dagger}$	$-15.50^{\dagger}$

† The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

Table 3 Comparison between NSDE, DE and FEP on $f_{14} - f_{23}$. All results have been averaged over 50 independent runs

Test Func	# of Gen's	NSDE Mean	DE Mean	# of Gen's	FEP Mean	vs DE t-test	vs FEP t-test
f_{14}	100	0.998	0.998	100	1.22	0	$-2.80^{\dagger}$
f_{15}	4000	3.07×10^{-04}	3.07×10^{-04}	4000	5.00×10^{-04}	0	$-4.26^{\dagger}$
f_{16}	100	−1.03	−1.03	100	−1.03	0	0
f_{17}	100	0.398	0.398	100	0.398	0	0
f_{18}	100	3.00	3.00	100	3.02	0	−1.29
f_{19}	100	−3.86	−3.86	100	−3.86	0	0
f_{20}	200	−3.32	−3.28	200	−3.27	$-4.97^{\dagger}$	$-5.99^{\dagger}$
f_{21}	100	−9.68	−10.15	100	−5.52	$4.58^{\dagger}$	$-16.83^{\dagger}$
f_{22}	100	−10.33	−10.40	100	−5.52	1.49	$-15.85^{\dagger}$
f_{23}	100	−10.48	−10.54	100	−6.57	$2.91^{\dagger}$	$-8.80^{\dagger}$

† The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

Table 4 Further experiments for NSDE on $f_{21} - f_{23}$. All results have been averaged over 50 independent runs

Test Func	# of Gen's	NSDE Mean	# of Gen's	DE Mean	FEP Mean	vs DE t-test	vs FEP t-test
f_{21}	200	−10.15	100	−10.15	−5.52	0	$-20.59^{\dagger}$
f_{22}	150	−10.40	100	−10.40	−5.52	0	$-16.28^{\dagger}$
f_{23}	150	−10.54	100	−10.54	−6.57	0	$-8.94^{\dagger}$

† The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test

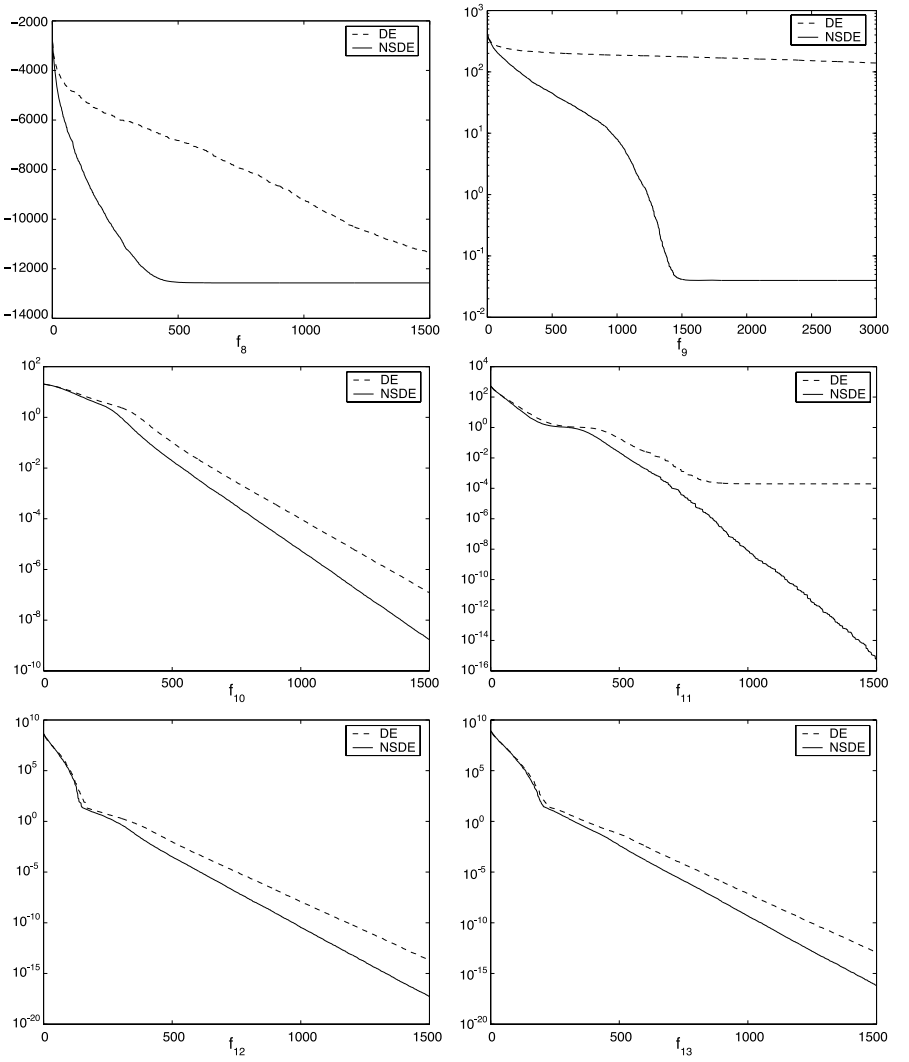


Fig. 3 Evolution process of the mean best values found for multimodal functions with many local optima, i.e. $f_8 - f_{13}$. The results were averaged over 50 runs. The *vertical axis* is the function value and the *horizontal axis* is the number of generations

convergence is a little slower than DE on these functions. The major difference between functions $f_8 - f_{13}$ and $f_{14} - f_{23}$ is that $f_{14} - f_{23}$ appears to be simpler than $f_8 - f_{13}$ due to their low dimensionalities and a smaller number of local optima [6]. But NSDE still spends some computational effort blindly to avoid being trapped in local optima when optimizing these functions. This will weaken the search strength in the direction towards the optimum. NSDE's performance indicates that the advantages of introducing neighborhood search become insignificant on this class of problems.

Scalability of an algorithm is also an important measurement of how good and how applicable the algorithm is [10]. So further experiments were conducted to evaluate NSDE's scalability against the growth of problem dimensions. We selected 9 scalable functions from the 23 benchmark functions. The dimensions D of them were set to 50, 100, 150 and 200, respectively. The computation times used by algorithms were set to grow in the order of $O(D)$ [10]. For example, for function f_1, 2500 generations were set for $D = 50$, 5000 for $D = 100$, 7500 for $D = 150$, and 10000 for $D = 200$. The average results of 50 independent runs are summarized in Tables 5 and 6.

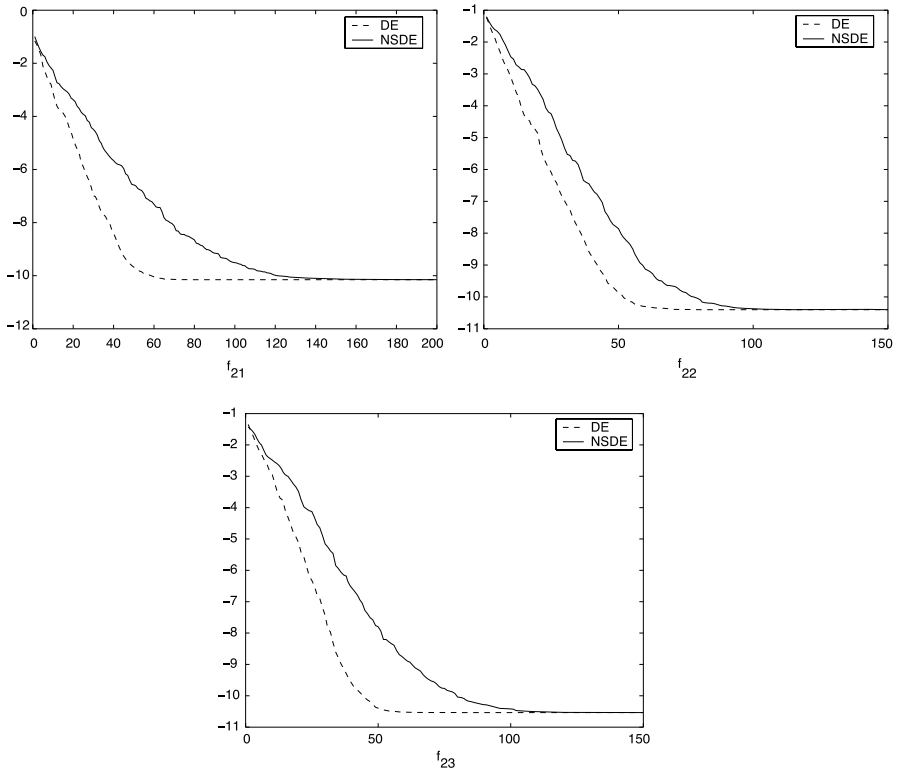


Fig. 4 The evolution process of the mean best values found for $f_{21} - f_{23}$ in further experiments. The results were averaged over 50 runs. The *vertical axis* is the function value and the *horizontal axis* is the number of generations

For unimodal functions, NSDE outperformed DE on $f_1 - f_3$ and f_6. The classical DE has only advantages on f_4 for dimensions 150 and 200. For functions $f_1 - f_3$, NSDE gained better and better results from 50-D to 200-D, with the computational time growth in the order of $O(D)$. But such good scalability was not observed for classical DE. Although DE appeared to achieve better results on the 100-D problem than the 50-D, its performance became poorer for problems of 150-D and 200-D. For function f_6, NSDE found the optimum from 50-D to 200-D, while DE only found the optimum for 50-D. For function f_4, both NSDE and DE performed worse and worse with the growth of dimensions. NSDE's performance decreased faster than DE. Function f_4's fitness value is determined by the maximum component of the D-dimensional vector (see the definition of f_4 [6]). Global evolutionary operator is needed to make progress when optimizing this function. Maybe the neighborhood search strategy in NSDE is too local to create better offspring for large scale f_4.

Table 6 shows the results for multimodal functions with many local optima. As mentioned in Sect. 4.1, this class of functions are often regarded as being difficult to optimize because the number of local optima increase exponentially as their dimension increases. It was encouraging to find NSDE outperformed DE on all of these functions, from 50-D to 200-D. For function f_8, NSDE's results were not only closer

Table 5 Comparison between NSDE and DE on $f_1 - f_6$, with dimension $D = 50, 100, 150$ and 200, respectively. All results have been averaged over 50 independent runs

Test	# of	# of	NSDE		DE		vs DE
Func	Dim's	Gen's	Mean	Std	Mean	Std	t-test
f_1	50	2500	4.28×10^{-18}	7.86×10^{-18}	5.48×10^{-15}	4.05×10^{-15}	$-9.56^{\dagger}$
	100	5000	2.07×10^{-22}	2.72×10^{-22}	2.91×10^{-17}	1.84×10^{-17}	$-11.18^{\dagger}$
	150	7500	1.58×10^{-25}	2.48×10^{-25}	4.82×10^{-17}	4.70×10^{-17}	$-7.25^{\dagger}$
	200	10,000	1.10×10^{-27}	1.47×10^{-27}	2.77×10^{-15}	5.26×10^{-15}	$-3.72^{\dagger}$
f_2	50	2500	2.02×10^{-11}	1.49×10^{-11}	1.05×10^{-07}	3.44×10^{-08}	$-21.58^{\dagger}$
	100	5000	1.58×10^{-13}	1.30×10^{-13}	7.45×10^{-10}	4.63×10^{-10}	$-11.38^{\dagger}$
	150	7500	2.27×10^{-15}	1.69×10^{-15}	2.13×10^{-09}	8.22×10^{-09}	$-1.83^{\dagger}$
	200	10,000	4.88×10^{-17}	3.49×10^{-17}	1.43×10^{-07}	6.50×10^{-07}	-1.56
f_3	50	2500	1.35×10^{-16}	2.44×10^{-16}	1.15×10^{-13}	8.34×10^{-14}	$-9.74^{\dagger}$
	100	5000	9.98×10^{-21}	1.55×10^{-20}	8.35×10^{-16}	6.79×10^{-16}	$-8.70^{\dagger}$
	150	7500	1.12×10^{-23}	1.34×10^{-23}	4.57×10^{-15}	4.58×10^{-15}	$-7.06^{\dagger}$
	200	10,000	1.08×10^{-25}	1.44×10^{-25}	1.89×10^{-13}	5.35×10^{-13}	$-2.50^{\dagger}$
f_4	50	2500	$1.12 \times 10^{+00}$	$1.85 \times 10^{+00}$	$4.86 \times 10^{+00}$	$2.26 \times 10^{+00}$	$-9.05^{\dagger}$
	100	5000	$2.00 \times 10^{+01}$	$6.10 \times 10^{+00}$	$2.26 \times 10^{+01}$	$4.44 \times 10^{+00}$	$-2.43^{\dagger}$
	150	7500	$3.34 \times 10^{+01}$	$4.83 \times 10^{+00}$	$3.09 \times 10^{+01}$	$3.95 \times 10^{+00}$	$2.81^{\dagger}$
	200	10,000	$4.31 \times 10^{+01}$	$6.20 \times 10^{+00}$	$3.41 \times 10^{+01}$	$3.32 \times 10^{+00}$	$8.95^{\dagger}$
f_6	50	2500	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	0.00
	100	5000	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	2.00×10^{-02}	1.41×10^{-01}	-1.00
	150	7500	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	4.00×10^{-02}	1.98×10^{-01}	-1.43
	200	10,000	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$4.02 \times 10^{+00}$	$1.34 \times 10^{+01}$	$-2.12^{\dagger}$

† The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test

Table 6 Comparison between NSDE and DE on $f_8 - f_{11}$, with dimension $D = 50, 100, 150$ and 200, respectively. All results have been averaged over 50 independent runs

Test Func	# of Dim's	# of Gen's	NSDE Mean	NSDE Std	DE Mean	DE Std	vs DE t-test
f_8	50	2500	− 20,946.80	16.70	− 15,928.00	3731.52	$-9.51^{\dagger}$
	100	5000	− 41,860.40	69.50	− 31,729.30	7533.67	$-9.51^{\dagger}$
	150	7500	− 62,568.70	216.56	− 54,961.90	5736.55	$-9.37^{\dagger}$
	200	10,000	− 83,044.50	348.88	− 73,964.00	2124.57	$-29.82^{\dagger}$
f_9	50	2500	6.57×10^{-01}	8.67×10^{-01}	$3.23 \times 10^{+02}$	$2.84 \times 10^{+01}$	$-80.31^{\dagger}$
	100	5000	$8.78 \times 10^{+00}$	$3.11 \times 10^{+00}$	$5.56 \times 10^{+02}$	$1.04 \times 10^{+02}$	$-37.18^{\dagger}$
	150	7500	$2.69 \times 10^{+01}$	$6.12 \times 10^{+00}$	$3.46 \times 10^{+02}$	$3.23 \times 10^{+02}$	$-6.99^{\dagger}$
	200	10,000	$5.30 \times 10^{+01}$	$1.05 \times 10^{+01}$	$1.61 \times 10^{+02}$	$1.69 \times 10^{+01}$	$-38.37^{\dagger}$
f_{10}	50	2500	3.74×10^{-10}	4.44×10^{-10}	1.64×10^{-08}	6.13×10^{-09}	$-18.44^{\dagger}$
	100	5000	1.91×10^{-12}	1.02×10^{-12}	8.59×10^{-10}	4.41×10^{-10}	$-13.74^{\dagger}$
	150	7500	4.81×10^{-14}	2.29×10^{-14}	3.09×10^{-01}	4.80×10^{-01}	$-4.55^{\dagger}$
	200	10,000	2.04×10^{-14}	3.53×10^{-15}	$1.48 \times 10^{+00}$	3.66×10^{-01}	$-28.59^{\dagger}$
f_{11}	50	2500	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	1.97×10^{-04}	1.39×10^{-03}	−1.00
	100	5000	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	5.92×10^{-04}	2.37×10^{-03}	$-1.77^{\dagger}$
	150	7500	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	1.13×10^{-03}	3.16×10^{-03}	$-2.53^{\dagger}$
	200	10,000	2.46×10^{-04}	1.74×10^{-03}	3.05×10^{-03}	8.14×10^{-03}	$-2.38^{\dagger}$

† The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

to optimum, but also more stable than classical DE (see the values of Std. Dev and t-test). For function f_9, NSDE's performance decreased a little as the dimensions increased, but was still much better than DE. For function f_{10}, NSDE showed similar good scalability as for unimodal functions, i.e., its results became better and better from 50-D to 200-D problems. For function f_{11}, NSDE found the optimum for 50-D, 100-D and 150-D problems, while DE never found the optimum for them. Although NSDE failed to find the optimum for 200-D f_{11}, it still outperformed DE significantly.

4.2 NSDE on CEC2005's Functions

To evaluate NSDE further, a new set of benchmark functions were used, including 25 functions with different complexity [11]. Many of them are the shifted, rotated, expanded or combined variants of classical functions. Functions $f_1 - f_5$ are unimodal while the remaining 20 functions are multimodal. The experimental results will compare with not only classical DE, but also another widely used algorithm, CMAES. DE and CMAES's experimental results were provided in [12, 13]. To be consistent with their experimental setting, experiments are conducted on all 25 $30 - D$ problems, and we chose the function evaluations (FEs) to be $3.0 \times 10^{+05}$. Error value, i.e. the difference between current fitness value and optimum value, is used to compare algorithm's performance. The average error values of 25 independent runs are summarized in Tables 7–9.

For unimodal functions $f_1 - f_5$, the three algorithms have comparable performance. NSDE and DE have exactly the same results on the Shifted Sphere function f_1. NSDE performed better on $f_2 - f_4$, but was outperformed by DE on f_5. It is remarkable that CMAES performed far better than NSDE on f_3 and f_5, but far worse on f_4. One possible reason is that CMAES and DE-based algorithms have very different search biases on these functions. Later we will trace the reason through characteristics of different functions.

For basic and expanded multimodal functions $f_6 - f_{14}$, the advantages of introducing neighborhood search are much more significant. In terms of experimental results, NSDE outperformed DE on almost all functions except f_7 and f_{12}. Although CMAES performed better on $f_6 - f_8$, and was outperformed by NSDE on the other six functions $f_9 - f_{14}$. NSDE is superior on these multimodal functions, which is consistent with the conclusions given for the classical functions.

After analyzing the characteristics of these functions, i.e. the first column of Tables 7 and 8, it is found that DE has rather poor performance on rotated or non-

Table 7 Comparison between NSDE, DE and CMAES on $f_1 - f_5$. All results have been averaged over 25 independent runs (S means the function is Shifted, R means Rotated, and N means Non-separable)

CEC'05 Func	NSDE Mean	DE Mean	CMAES Mean	vs DE t-test	vs CMAES t-test
f_1(S/-/-)	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	5.28×10^{-09}	0	$-26.9^{\dagger}$
f_2(S/-/N)	5.62×10^{-08}	3.33×10^{-02}	6.93×10^{-09}	$-3.40^{\dagger}$	$3.78^{\dagger}$
f_3(S/R/N)	$6.40 \times 10^{+05}$	$6.92 \times 10^{+05}$	5.18×10^{-09}	$-74.9^{\dagger}$	$11.4^{\dagger}$
f_4(S/-/N)	$9.02 \times 10^{+00}$	$1.52 \times 10^{+01}$	$9.26 \times 10^{+07}$	-1.41	$-2.76^{\dagger}$
f_5(-/-/N)	$1.56 \times 10^{+03}$	$1.70 \times 10^{+02}$	8.30×10^{-09}	$15.3^{\dagger}$	$18.8^{\dagger}$

† The value of t with 24 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test

Table 8 Comparison between NSDE, DE and CMAES on $f_6 - f_{14}$. All results have been averaged over 25 independent runs (S means the function is Shifted, R means Rotated, and N means Non-separable)

CEC'05 Func	NSDE Mean	DE Mean	CMAES Mean	vs DE t-test	vs CMAES t-test
f_6(S/-/N)	$2.45 \times 10^{+01}$	$2.51 \times 10^{+01}$	6.31×10^{-09}	$-7.60^{\dagger}$	$4.57^{\dagger}$
f_7(S/R/N)	1.18×10^{-02}	2.96×10^{-03}	6.48×10^{-09}	$3.41^{\dagger}$	$5.04^{\dagger}$
f_8(S/R/N)	$2.09 \times 10^{+01}$	$2.10 \times 10^{+01}$	$2.00 \times 10^{+01}$	$-6.42^{\dagger}$	$76.7^{\dagger}$
f_9(S/-/-)	7.96×10^{-02}	$1.85 \times 10^{+01}$	$2.91 \times 10^{+02}$	$-1.77^{\dagger}$	$-41.1^{\dagger}$
f_{10}(S/R/N)	$4.29 \times 10^{+01}$	$9.69 \times 10^{+01}$	$5.63 \times 10^{+02}$	$-3.22^{\dagger}$	$-10.5^{\dagger}$
f_{11}(S/R/N)	$1.41 \times 10^{+01}$	$3.42 \times 10^{+01}$	$1.52 \times 10^{+01}$	$-7.28^{\dagger}$	-0.56
f_{12}(S/-/N)	$6.59 \times 10^{+03}$	$2.75 \times 10^{+03}$	$1.32 \times 10^{+04}$	$2.76^{\dagger}$	$-2.53^{\dagger}$
f_{13}(S/-/N)	$1.62 \times 10^{+00}$	$3.23 \times 10^{+00}$	$2.32 \times 10^{+00}$	$-9.25^{\dagger}$	$-7.84^{\dagger}$
f_{14}(S/R/N)	$1.32 \times 10^{+01}$	$1.34 \times 10^{+01}$	$1.40 \times 10^{+01}$	$-5.14^{\dagger}$	$-9.40^{\dagger}$

† The value of t with 24 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test

Table 9 Comparison between NSDE, DE and CMAES on $f_{15} - f_{25}$. All results have been averaged over 25 independent runs

CEC'05 Func	NSDE Mean	DE Mean	CMAES Mean	vs DE t-test	vs CMAES t-test
f_{15}	$3.64 \times 10^{+02}$	$3.60 \times 10^{+02}$	$2.16 \times 10^{+02}$	0.15	$6.59^{\dagger}$
f_{16}	$6.90 \times 10^{+01}$	$2.12 \times 10^{+02}$	$5.84 \times 10^{+01}$	$-6.26^{\dagger}$	1.68
f_{17}	$1.01 \times 10^{+02}$	$2.37 \times 10^{+02}$	$1.07 \times 10^{+03}$	$-5.15^{\dagger}$	$-9.40^{\dagger}$
f_{18}	$9.04 \times 10^{+02}$	$9.04 \times 10^{+02}$	$8.90 \times 10^{+02}$	0	1.52
f_{19}	$9.04 \times 10^{+02}$	$9.04 \times 10^{+02}$	$9.03 \times 10^{+02}$	0	0.61
f_{20}	$9.04 \times 10^{+02}$	$9.04 \times 10^{+02}$	$8.89 \times 10^{+02}$	0	1.65
f_{21}	$5.00 \times 10^{+02}$	$5.00 \times 10^{+02}$	$4.85 \times 10^{+02}$	0	$2.21^{\dagger}$
f_{22}	$8.89 \times 10^{+02}$	$8.97 \times 10^{+02}$	$8.71 \times 10^{+02}$	$-2.00^{\dagger}$	$3.43^{\dagger}$
f_{23}	$5.34 \times 10^{+02}$	$5.34 \times 10^{+02}$	$5.35 \times 10^{+02}$	0	$-3.27^{\dagger}$
f_{24}	$2.00 \times 10^{+02}$	$2.00 \times 10^{+02}$	$1.41 \times 10^{+03}$	0	$-11.1^{\dagger}$
f_{25}	$2.00 \times 10^{+02}$	$7.30 \times 10^{+02}$	$6.91 \times 10^{+02}$	$-7.09 \times 10^{+03\dagger}$	$-3.12^{\dagger}$

† The value of t with 24 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

separable functions. CMAES has superior performance on f_2, f_3 and $f_5 - f_7$, but still gained poor results on the remaining functions. As an improved version of DE, although NSDE outperformed DE on some of these rotated or non-separable functions, the results of f_3, $f_5 - f_8$ and f_{12} are still unsatisfactory. Despite success in expanding the neighborhood search ability, NSDE's performance is still limited by the inherited framework of original DE.

For hybrid composition functions $f_{15} - f_{25}$, the results in Table 9 show that all algorithms not only failed to locate the optimum, but also become trapped in local optima that are far from optimum. These functions are much more difficult, and no effective algorithms have yet been found to slve them [14]. A more detaile investigation of the results shows that NSDE still outperformed DE on f_{17}, f_{22}, f_{25}, and outperformed CMAES on f_{17}, $f_{23} - f_{25}$. These functions are the hybrid composition of basic rotated or non-separable functions. The reason why NSDE is inefficient on some of these functions is as found in the analysis of $f_1 - f_{14}$, i.e. NSDE's performance on non-separable functions is limited by the inherited framework of DE. It is interesting to note that NSDE's results on f_{17} and f_{25} are much closer to the optimum than those of DE and CMAES. The strategy of introducing large jumps to escape from local optima in NSDE is still useful even on some of these composition multimodal functions.

5 Conclusions

This chapter proposes NSDE , an improved variation of classical DE, which is inspired by EP's neighborhood search (NS) strategy. Gaussian and Cauchy NS operators are introduced into NSDE. The advantages of DE with neighborhood search are analyzed theoretically. It has been shown that NS operators will improve the diversity of DE's

search step size and population, which will be beneficial to escape from local optima when searching in environments without prior knowledge of what search step size is prefered.

Experimental evidence is also provided showing how the neighborhood search (NS) strategy affects DE's evolutionary behavior. A total of 48 widely used benchmark problems were employed to test NSDE's performance. Our experimental results show that DE with neighborhood search has significant advantages over classical DE.

Acknowledgement. The authors are grateful to Prof. P.N. Suganthan and Dr. Tang Ke for their constructive comments on this chapter. This work is partially supported by the National Science Foundation of China (Grant No. 60428202 and 60573170).

References

1. R. Storn, K. Price (1997) Differential Evolution – A Simple and Efficient Heuristic Strategy for Global Optimization over Continuous Spaces. Journal of Global Optimization, 11:341–359
2. R. Thomsen (2003) Flexible Ligand Docking using Differential Evolution. Proc. of the 2003 Congress on Evolutionary Computation, 4:2354–2361
3. A.K. Qin, P.N. Suganthan (2005) Self-adaptive Differential Evolution Algorithm for Numerical Optimization. Proc. of the 2005 Congress on Evolutionary Computation, 2:1785–1791
4. D. Zaharie (2002) Critical Values for the Control Parameters of Differential Evolution Algorithms. Proc. of Mendel 2002, 8th International Conference on Soft Computing, 62–67
5. R. Gämperle, S. D. Müller, P. Koumoutsakos (2002) A Parameter Study for Differential Evolution. Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, 293–298
6. X. Yao, Y. Liu, G. Lin (1999) Evolutionary Programming Made Faster. IEEE Transactions on Evolutionary Computation, 3:2:82–102
7. C. Lee, X. Yao (2004) Evolutionary Programming Using Mutations Based on the Lévy Probability Distribution. IEEE Transactions on Evolutionary Computation, 8:1:1–13
8. T. Bäck, H. P. Schwefel (1993) An Overview of Evolutionary Algorithms for Parameter Optimization. Evolutionary Computation, 1:1–23
9. J. Vesterstrom, R. Thomsen (2004) A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems. Evolutionary Computation, 2:1980–1987
10. Y. Liu, X. Yao, Q. Zhao, T. Higuchi (2001) Scaling Up Fast Evolutionary Programming with Cooperative Coevolution. Proc. of the 2001 Congress on Evolutionary Computation, 1:1101–1108
11. P. N. Suganthan et al. (2005) Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. http://www.ntu.edu.sg/home/EPNSugan
12. J. Rönkkönen, S. Kukkonen, K. V. Price (2005) Real-Parameter Optimization with Differential Evolution. Proc. of the 2005 Congress on Evolutionary Computation, 1:567–574

13. A. Auger, S. Kern, N. Hansen (2005) Performance Evoluation of an Advanced Local Search Evolutionary Algorithm. Proc. of the 2005 Congress on Evolutionary Computation, 2:1777–1784
14. H. Nikolaus (2005) Compilation of Results on the 2005 CEC Benchmark Function Set. http://www.ntu.edu.sg/home/EPNSugan