Differential Evolution Enhanced by Neighborhood Search

Hui Wang, Zhijian Wu and Shahryar Rahnamayan

Abstract— **This paper presents a novel Differential Evolution (DE) algorithm, called DE enhanced by neighborhood search (DENS), which differs from pervious works of utilizing the neighborhood search in DE, such as DE with neighborhood search (NSDE) and self-adaptive DE with neighborhood search (SaNSDE). In DENS, we focus on searching the neighbors of individuals, while the latter two algorithms (NSDE and SaNSDE) work on the adaption of the control parameters F and CR. The proposed algorithm consists of two following main steps. First, for each individual, we create two trial individuals by local and global neighborhood search strategies. Second, we select the fittest one among the current individual and the two created trial individuals as a new current individual. Experimental studies on a comprehensive set of benchmark functions show that DENS achieves better results for a majority of test cases, when comparing with some other similar evolutionary algorithms.**

Index Terms— **Differential evolution, neighborhood search, local search, global optimization.**

I. Introduction

Differential Evolution (DE), proposed by Price and Storn [1], is an effective, robust, and simple global optimization algorithm. According to frequently reported experimental studies, DE has shown better performance than many other evolutionary algorithm (EAs) in terms of convergence speed and robustness over several benchmark functions and real-world problems [2].

Since the development of DE, many improved versions have been proposed. Based on the improved mechanisms, we can divide them into three categories as follows.

1) *Adaptive Parameter Control*: The classical DE algorithm only has three control parameters N_p (population size), CR and F, which greatly affect performance of DE. The values of these parameters highly determine the quality of the obtained solution and the efficiency of the search [3]. Choosing appropriate parameter values is a problem dependent task and requires previous experience and knowledge of the user. To tackle this problem, some adaptive parameter control strategies have been proposed, such as fuzzy DE (FADE) [4] self-adaptive DE (SaDE) [5], [6], self-adapting control parameters in DE (jDE) [3], DE with neighborhood search (NSDE) [7] and self-adaptive DE with neighborhood search (SaNSDE) [8].
2) *Modified Mutation Strategies*: The DE algorithm has two important operators (besides the selection), mutation and crossover. The former is determined by the mutation strategies, and the latter is dominated by the crossover probability CR and crossover strategy. Besides the improvement of the control parameters, some modifications of the mutation strategies could also improve the performance of DE. DE/current-to-*p*best [9] and DE/target-to-best/1 [10] are some examples among others.
3) *Hybrid Strategies*: Recently, some new works have been introduced by combining the classical DE with the concepts of machine learning and some successful search techniques. These new improved DE variants are called hybrid DE, such as opposition-based DE (ODE) [11], [12], [13], [14], DE with adaptive local search (DEahcSPX) [15], and DE based on generalized opposition-based learning (GODE) [16].

In this paper, we present a novel DE algorithm, called DE enhanced by neighborhood search (DENS), to improve the performance of the standard DE. In order to verify the performance of DENS, current work provides a comparative study of DENS and other similar DE variants on a comprehensive set of benchmark functions.

The rest of the paper is organized as follows. In Section II, the classical DE algorithm is briefly reviewed. The proposed approach, DENS, is presented in Section III. In Section IV, the test functions, parameter settings and the comparison of DENS with other similar algorithms are provided. Finally, the work is summarized and concluded in Section V.

II. A Brief Review of Differential Evolution

DE is a population-based stochastic search algorithm, and has been successfully applied to solve complex problems including linear and nonlinear, unimodal and multimodal functions. It has been investigated that DE is faster and more robust on majority of functions than many other evolutionary algorithms [2].

There are several variants of DE [1], where the most popular variant is indicated by "$DE/rand/1/bin$" which is called classical version. The proposed algorithm is also based on this DE scheme. Let us assume that $X_{i,G}(i = 1, 2, \ldots, N_p)$ is the ith individual in population $P(G)$, where N_p is the population size, G is the generation index, and $P(G)$ is the population in the Gth generation. The main idea of DE is to generate trial vectors. Mutation and crossover are used to produce new trial vectors, and selection determines which of the vectors will be successfully selected into the next generation.

Hui Wang and Zhijian Wu are with the State Key Laboratory of Software Engineering, Wuhan University, Wuhan, 430072 China (e-mail: wanghui_cug@yahoo.com.cn; zjwu9551@sina.com).

Shahryar Rahnamayan is with Faculty of Engineering and Applied Science, University of Ontario Institute of Technology (UOIT), 2000 Simcoe Street North, Oshawa, ON L1H 7K4, Canada (e-mail: shahryar.rahnamayan@uoit.ca).

Mutation–For each vector $X_{i,G}$ in generation G, a mutant vector V is generated by

$$V_{i,G} = X_{i_1,G} + F\left(X_{i_2,G} - X_{i_3,G}\right), \quad (1)$$
$$i \neq i_1 \neq i_2 \neq i_3,$$

where $i = 1, 2, \ldots, N_p$ and i_1, i_2, and i_3 are mutually different random integer indices within $\{1, 2, \cdots, N_p\}$. The population size N_p should be satisfied by $N_p \geq 4$ because i, i_1, i_2, and i_3 are different. $F \in [0, 2]$ is a real number that controls the amplification of the difference vector $(X_{i_2,G} - X_{i_3,G})$.

Crossover–Similar to genetic algorithms, DE also employs a crossover operator to build trial vectors by recombining two different vectors. The trial vector is defined as follows:

$$U_{i,G} = \left(U_{i,1,G}, U_{i,2,G}, \ldots, U_{i,D,G}\right), \quad (2)$$

where $j = 1, 2, \ldots, D$ and

$$U_{i,j,G} = \begin{cases} V_{i,j,G}, & \text{if } rand_j(0,1) \leq CR \vee j = l \\ X_{i,j,G}, & \text{otherwise} \end{cases}. \quad (3)$$

$CR \in (0, 1)$ is the predefined crossover probability, and $rand_j(0, 1)$ is a random number within $[0, 1]$ for the jth dimension, and $l \in \{1, 2, \ldots, D\}$ is a random parameter index.

Selection–A greedy selection mechanism is used as follows:

$$X_{i,G} = \begin{cases} U_{i,G}, & \text{if } f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G}, & \text{otherwise} \end{cases}. \quad (4)$$

Without loss of generality, this paper only considers minimization problem. If, and only if, the trial vector $U_{i,G}$ is better than $X_{i,G}$, then $X_{i,G}$ is set to $U_{i,G}$; otherwise, the $X_{i,G}$ is unchanged.

III. DE Enhanced by Neighborhood Search

A. Literature Review

Like other stochastic algorithms, DE also suffers from the problem of premature convergence when solving complex multimodal problems. Sometimes, the suboptimum is near to the global optimum and the neighborhoods of trapped individuals may cover the global optimum. At such situation, searching the neighborhood of an individual is helpful to find better solutions. In this paper, we propose a hybrid DE algorithm, called DENS, to search the neighborhoods of individuals. The proposed approach differs from previous neighborhood search strategies in DE. Before introducing the DENS, we need to give a brief review of other DE variants equipped by neighborhood search.

Yang et al. [7] introduced a neighborhood search strategy for DE (NSDE), which generates F using Gaussian and Cauchy distributed random numbers instead of predefining a constant F. In NSDE, different values of F indicate the different mutant vectors in the neighborhood of current vector. Based on SaDE [5] and NSDE, Yang et al. [8] proposed another version of DE, called self-adaptive DE with neighborhood search (SaNSDE), which inherits from NSDE to generate self-adaptive F, and uses a weighted adaptation scheme for CR. The presented experimental results show that SaNSDE outperforms SaDE and NSDE. As seen, the above two DE variants with neighborhood search focus on the adaption of the control parameters.

$$\begin{aligned} V_{i,G} =& X_{i,G} + F \cdot (X_{best,G} - X_{i,G}) \\ &+ F \cdot (X_{r_1,G} - X_{r_2,G}), \end{aligned} \quad (5)$$

where $X_{best,G}$ indicates the best vector in the population at generation G, $r_1, r_2 \in \{1, 2, \cdots, N_p\}$, and $i \neq r_1 \neq r_2$.

There is a tradeoff between *exploration* and *exploitation* in most of evolutionary algorithms (EAs). The former indicates the local search ability and makes the algorithm explore every region of the feasible search space, while the latter means the global search ability and accelerates the algorithm converging to the near-optimal solutions. Most improvements on EAs try to seek a balance these two factors that suits the different kinds of problems. The $DE/target-to-best/1$ strategy described in Eq.5 promotes *exploitation* sine all the vectors move to the same best position by the attraction of X_{best}, thereby converging faster to that point [10]. But in many cases, the population may lose its global exploration abilities within a relatively small number of generations, thereafter getting trapped to some locally optimal point in the search space. To tackle this problem, Das et al. [10] proposed an enhanced DE algorithm (DEGL) by using an improved $DE/target-to-best/1$ strategy which includes two mutation strategies: local neighborhood and global neighborhood.

Local Neighborhood Mutation–In the local model, each vector is mutated using the best position found so far in a small neighborhood of it and not the whole population. Thereby, the vectors are no longer attracted by the same point. The modified model is defined by

$$\begin{aligned} L_{i,G} =& X_{i,G} + \alpha \cdot (X_{n\_best_i,G} - X_{i,G}) \\ &+ \beta \cdot (X_{p,G} - X_{q,G}), \end{aligned} \quad (6)$$

where the subscript n_best_i indicates the best vector in the neighborhood of $X_{i,G}$, $p, q \in [i-k, i+k]$ with $p \neq q \neq i$, and k is the neighborhood size. The vectors $X_{n\_best_i,G}$, $X_{p,G}$ and $X_{q,G}$ are defined on a small neighborhood of $X_{i,G}$, and the search behavior of each vector is almost independent. The information of vectors spread through the population regarding the best position of each neighborhood. Therefore, the attraction to specific points is weaker, which prevents the population from getting trapped into local minima [10].

Global Neighborhood Mutation–Besides the local neighborhood mutation, the DEGL also employs a global neighborhood model by adding two scaling factors α and β in the original $DE/target-to-best/1$ strategy as follows.

$$G_{i,G} = X_{i,G} + \alpha \cdot (X_{best,G} - X_{i,G}) + \beta \cdot (X_{r_1,G} - X_{r_2,G}), \quad (7)$$

where the subscript X_{best} indicates the best vector in the entire population at generation G, $r_1, r_2 \in \{1, 2, \cdots, N_p\}$ with $r_1 \neq r_2 \neq i$, and N_p is the population size. The parameters α and β are the scaling factors.

Based on the two neighborhood mutations, DEGL combines them using a scalar weight $w \in (0,1)$ to form a new mutation strategy instead of the original $DE/rand/1/bin$ or $DE/target-to-best/1$ strategy.

$$V_{i,G} = w \cdot G_{i,G} + (1-w) \cdot L_{i,G}. \quad (8)$$

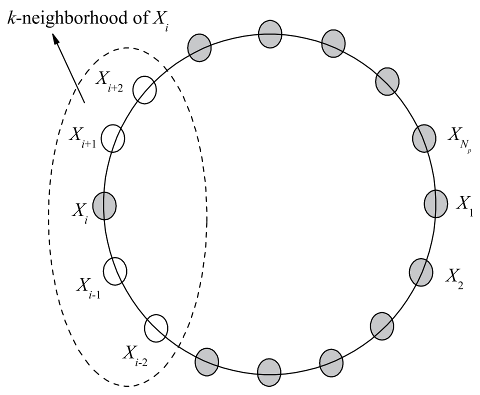


Fig. 1. The k-neighborhood in a ring topology, where $k = 2$.

B. The Proposed Approach

In the DEGL, a static ring topology of neighborhood is defined on the set of indices of the vectors. The vector $X_{i,G}$ is connected by $X_{i+1,G}$ and $X_{i-1,G}$. For instance, $X_{2,G}$ and $X_{N_p,G}$ are two immediate neighbors of $X_{1,G}$. On the basis of the ring topology, DEGL defines a k-neighborhood radius in the ring topology, consisting of vectors $X_{i-k,G}, \ldots, X_{i,G}, \ldots, X_{i+k,G}$, for each X_i, where k is an integer within $\{0, 1, \cdots, \frac{N_p - 1}{2}\}$, as the neighborhood size must be smaller than the population size $2k+1 \leq N_p$. Fig. 1 presents the k-neighborhood radius, where $k = 2$. In the local neighborhood mutation, DEGL selects the best vectors and two random vectors in the k-neighborhood radius of $X_{i,G}$.

However, the above selection range is not the real neighborhood of the current vector $X_{i,G}$, but the entire population. Because the ring topology is defined on the indices of the vectors, but not the Euclidean distances among the vectors. The immediate neighbors $X_{i+1,G}$ and $X_{i-1,G}$ of $X_{i,G}$ may not the nearest neighbor to $X_{i,G}$. In Eq.6, the $X_{n\_best_i,G}$, $X_{p,G}$ and $X_{q,G}$ are not the nearest neighbors to $X_{i,G}$ in the whole population.

Algorithm 1: DE Enhanced by Neighborhood Search (DENS).

```
1  Randomly initialize each individual in the population $P(G)$;
2  Calculate the fitness value of each $X_{i,G}$;
3  NE = $N_p$;
4  Initialize $X_{pbest_i,G}$ and $X_{best,G}$;
5  while $NE \leq MAX_{NE}$ do
       /* Execute the classical DE */
6      for $i = 1$ to $N_p$ do
7          Randomly select 3 vectors $X_{i_1,G}$, $X_{i_2,G}$ and $X_{i_3,G}$
           from $P(G)$, where $i \neq i_1 \neq i_2 \neq i_3$ ;
8          $V_{i,G} = X_{i_1,G} + F(X_{i_2,G} - X_{i_3,G})$;
9          for $j = 1$ to $D$ do
10             if $rand(0,1) < CR$ then
11                 $U_{i,j,G} = V_{i,j,G}$;
12             end
13             else
14                 $U_{i,j,G} = X_{i,j,G}$;
15             end
16         end
17         Calculate the fitness value of $U_{i,G}$;
18         NE = NE + 1;
19         if $f(U_{i,G}) \leq f(X_{i,G})$ then
20             $X_{i,G} = U_{i,G}$
21         end
22         Update $X_{pbest_i,G}$ and $X_{best,G}$, if needed;
23     end
       /* Conduct the neighborhood search */
24     for $i = 1$ to $N_p$ do
25         if $rand(0,1) \leq p_{ns}$ then
26             Create two trial vectors $L_{i,G}$ and $G_{i,G}$ according
               to Eq.11 and Eq.12, respectively;
27             Calculate the fitness values of $L_{i,G}$ and $G_{i,G}$;
28             NE = NE + 2;
29             Select the fittest vectors from $\{X_{i,G}$, $L_{i,G}$ and
               $G_{i,G}\}$ as new $X_{i,G}$;
30         end
31         $X_{i,G+1} = X_{i,G}$;
32     end
33     $G = G + 1$;
34 end
```

In this paper, we propose another neighborhood search scheme which is inspired from the basic idea of DEGL [10] and also particle swarm optimization (PSO) [17]. In PSO, particles are attracted by their previous best particles and the global best particle. Whenever a particle flies towards good points in the search space, it continuously modifies its trajectory by learning its previous best particle and the global best particle. Both $DE/target-to-best/1$ and DEGL only inherit from the experiences of the global best vector. In our approach, a vector not only learns the exemplar of its previous best vector X_{pbest_i}, but also learns from the experience of the global best vector X_{best}. As mentioned before, the defined k-neighborhood radius does not really indicate the nearest neighbors to the current vector. So we select the $X_{p,g}$ and $X_{q,g}$ in the whole population to simplify the operation. The modified local neighborhood strategy is

TABLE I

THE AVERAGE RESULTS ACHIEVED BY DENS WITH DIFFERENT VALUES OF p_{ns} ON BENCHMARK 1

Functions	$p_{ns} = 0.0$ (DE) Mean	$p_{ns} = 0.05$ Mean	$p_{ns} = 0.15$ Mean	$p_{ns} = 0.35$ Mean	$p_{ns} = 0.55$ Mean	$p_{ns} = 1.0$ Mean
f_1	5.88e–16	2.81e–62	4.71e–105	3.83e–150	9.82e–176	**2.49e–212**
f_2	2.37e–08	2.92e–31	9.36e–53	2.65e–65	3.46e–88	**4.91e–106**
f_3	0.56	1.35e–38	1.21e–77	1.45e–122	7.22e–149	**1.92e–188**
f_4	1.95	1.39e–26	2.98e–47	2.04e–69	1.26e–83	**5.39e–102**
f_5	18.6	**17.1**	19.6	21.8	23.9	26.1
f_6	**0**	**0**	**0**	**0**	**0**	**0**
f_7	6.08e–03	8.04e–04	2.89e–04	9.56e–05	**4.08e–05**	5.70e–05
f_8	–6120.2	–6639.5	–6824.6	–6695.9	–6726.2	**–6839.6**
f_9	173.4	**0**	**0**	**0**	**0**	**0**
f_{10}	5.97e–09	**5.89e–16**	**5.89e–16**	**5.89e–16**	**5.89e–16**	**5.89e–16**
f_{11}	2.09e–15	**0**	**0**	**0**	**0**	**0**
f_{12}	5.12e–17	**3.07e–17**	8.07e–17	3.89e–14	5.94e–12	2.28e–09
f_{13}	3.27e–16	**5.45e–17**	2.19e–03	1.79e–12	2.20e–03	6.12e–02
f_{14}	**0.98804**	**0.98804**	**0.98804**	**0.98804**	**0.98804**	**0.98804**
f_{15}	**3.07e–04**	**3.07e–04**	**3.07e–04**	**3.07e–04**	**3.07e–04**	**3.07e–04**
f_{16}	**–1.03163**	**3.07e–04**	**3.07e–04**	**3.07e–04**	**3.07e–04**	**3.07e–04**
f_{17}	**0.3979**	**0.3979**	**0.3979**	**0.3979**	**0.3979**	**0.3979**
f_{18}	**3**	**3**	**3**	**3**	**3**	**3**
f_{19}	**–10.15**	**–10.15**	**–10.15**	**–10.15**	**–10.15**	**–10.15**
f_{20}	**–10.40**	**–10.40**	**–10.40**	**–10.40**	**–10.40**	**–10.40**
f_{21}	**–10.54**	**–10.54**	**–10.54**	**–10.54**	**–10.54**	**–10.54**

defined by

$$\begin{aligned} L_{i,G} =& X_{i,G} + \alpha \cdot (X_{pbest_i,G} - X_{i,G}) \\ & + \beta \cdot (X_{p,G} - X_{q,G}), \end{aligned} \qquad (9)$$

where $X_{pbest_i,G}$ is the previous best vector of $X_{i,G}$ at generation G, p and q are two random integers within $\{1, 2, \cdots, N_p\}$.

The Eq.9 can be rewritten by

$$\begin{aligned} L_{i,G} =& (1-\alpha) \cdot X_{i,G} + \alpha \cdot X_{pbest_i,G} \\ & + \beta \cdot (X_{p,G} - X_{q,G}). \end{aligned} \qquad (10)$$

To simply the scaling factors $(1 - \alpha)$, α and β in Eq.10, we use three random numbers a_1, a_2 and a_3 instead of them, where $a_1, a_2, a_3 \in [0, 1]$ and $a_1 + a_2 + a_3 = 1$. Then, we get a new local neighborhood model as follows.

$$\begin{aligned} L_{i,G} =& a_1 \cdot X_{i,G} + a_2 \cdot X_{pbest_i,G} \\ & + a_3 \cdot (X_{p,G} - X_{q,G}). \end{aligned} \qquad (11)$$

Similar to the local model, we define the global neighborhood model as follows.

$$\begin{aligned} G_{i,G} =& a_1 \cdot X_{i,G} + a_2 \cdot X_{best,G} \\ & + a_3 \cdot (X_{r_1,G} - X_{r_2,G}), \end{aligned} \qquad (12)$$

where $X_{best,G}$ indicates the global best vector in the entire population at generation G, $r_1, r_2 \in \{1, 2, \cdots, N_p\}$ with $r_1 \neq r_2 \neq i$. The random numbers a_1, a_2 and a_3 are the same for each $X_{i,G}$, and they are generated anew in each generation.

In the proposed approach, DENS, we use two modified neighborhood search strategies (Eq.11 and Eq.12) to create two trial vectors $L_{i,G}$ and $G_{i,G}$ around the current vector $X_{i,G}$. And then, the fittest one among $X_{i,G}$, $L_{i,G}$ and $G_{i,G}$ is selected as the new $X_{i,G}$. The main steps of the DENS are described in Algorithm 1, where $X_{pbest_i,G}$ is the previous best vector of $X_{i,G}$, $X_{best,G}$ is the global best vector found so far in the population, G indicates the generation index, p_{ns} is the probability of the neighborhood search, NE is the number of function evaluations, and MAX_{NE} is the maximum number of function evaluations.

IV. EXPERIMENTAL VERIFICATIONS

A. Benchmark Functions

Experimental verifications for the proposed DENS are conducted on two sets of benchmark functions. The first benchmark (**Benchmark 1**) includes 21 well-known classical functions, which were used in [3]. The second test suit (**Benchmark 2**) includes 10 functions, which were provided in CEC 2005 special session [18]. All the functions used in this paper are minimization problems.

B. Results on ***Benchmark 1***

The **Benchmark 1** includes 21 functions, in which $f_1 - f_{13}$ are with dimension of 30 and $f_{14} - f_{21}$ are low-dimensional functions. Functions f_1-f_5 are unimodal; function f_6 is a step function, which has one minimum and is discontinuous; function f_7 is a noisy function. Functions f_8-f_{13} are multimodal functions where the number of local minima increases exponentially with the problem dimension [19]. Functions f_{14}-f_{21} are low-dimensional problems which have only a

TABLE II

THE COMPARISON OF DENS WITH SADE, NSDE AND SANSDE ON BENCHMARK 1

Functions	Dimension	MAX$_{\text{NE}}$	DE Mean	SaDE Mean	NSDE Mean	SaNSDE Mean	DENS Mean
f_1	30	150000	1.48e–25	7.49e–20	7.76e–20	3.02e–23	**2.01e–95**
f_2	30	150000	1.63e–13	6.22e–11	4.51e–10	4.64e–11	9**.18e–48**
f_3	30	150000	2.40e–03	1.12e–18	1.06e–14	6.62e–22	**1.54e–61**
f_4	30	150000	1.68	2.96e–02	2.54e–02	1.59e–03	**5.04e–40**
f_5	30	500000	1.09e–22	2.10e+01	1.24e+01	**4.13e–30**	8.89e–21
f_6	30	150000	**0**	**0**	**0**	**0**	**0**
f_7	30	150000	3.45e–03	7.58e–03	1.20e–02	7.21e–03	**5.23e–04**
f_8	30	150000	–6603.5	**–12569.5**	**–12569.5**	**–12569.5**	–7519.8
f_9	30	150000	142.86	4.00e–08	7.97e–02	1.84e–05	**0**
f_{10}	30	150000	7.95e–14	9.06e–11	6.72e–09	2.36e–12	**5.89e–16**
f_{11}	30	150000	**0**	8.88e–18	6.72e–09	**0**	**0**
f_{12}	30	150000	3.02e–17	1.21e–19	5.63e–17	**5.94e–23**	3.02e–17
f_{13}	30	150000	2.88e–17	1.75e–19	5.52e–16	**3.12e–22**	2.88e–17
f_{14}	2	20000	**0.998**	**0.998**	**0.998**	**0.998**	**0.998**
f_{15}	4	150000	**3.07e–04**	**3.07e–04**	**3.07e–04**	**3.07e–04**	**3.07e–04**
f_{16}	2	20000	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**
f_{17}	2	20000	**0.398**	**0.398**	**0.398**	**0.398**	**0.398**
f_{18}	2	20000	**3**	**3**	**3**	**3**	**3**
f_{19}	4	20000	**–10.15**	**–10.15**	**–10.15**	**–10.15**	**–10.15**
f_{20}	4	20000	**–10.40**	**–10.40**	**–10.40**	**–10.40**	**–10.40**
f_{21}	4	20000	**–10.54**	**–10.54**	**–10.54**	**–10.54**	**–10.54**

few local minima [19]. More details about the definitions of the **Benchmark 1** can be found in Table I in [3].

1) The Results of DENS with Different Values of p_{ns}:

In this subsection, we analyze the effects of p_{ns} on the performance of DENS. To accomplish the task, different values of p_{ns} are tested. The p_{ns} is set to 0.0, 0.05, 0.15, 0.35, 0.55 and 1.0 in the experiments, respectively. That will help to select a better value for p_{ns}.

The parameter settings in this experiment are described as follows. The population size, N_p, is set to 100 [3], [13]. The control parameters F and CR are set to 0.5 and 0.9 [13], respectively. The mutation strategy used in DENS is $DE/rand/1/bin$. The maximum number of functions evaluations MAX$_{\text{NE}}$ is set to 100,000 for all functions.

The results for DENS with different values of p_{ns} are presented in Table I, where "Mean" indicates the mean best function values found in the last generation.

From the results of Table I, it can be seen that a larger p_{ns} shows better performance on unimodal functions except for f_5 and f_6, while a smaller p_{ns} ($p_{ns} > 0$) achieves better results on multimodal functions f_{12} and f_{13}. For the rest multimodal functions, DENS with different p_{ns} almost obtains the same results. When $p_{ns} = 0.0$, in fact, DENS is equal to the classical DE, and the neighborhood search does not occur at all. Under this case, the algorithm performs very poor and fails into local minima on many test functions, while the algorithm just with few neighborhood searches ($p_{ns} = 0.05$) works much better. It shows that the neighborhood search is very helpful to improve the performance of DE. However, it is not possible to select the best p_{ns} in DENS, because that is problem oriented so that the probability of the neighborhood search should be empirically adjusted for different functions. In the DENS, p_{ns} with value 0.05 is regarded as the relatively suitable one for all test functions.

2) The Comparison of DENS with DE, SaDE, NSDE and SaNSDE:

In this subsection, we present a comparative study of DE, SaDE, NSDE, SaNSDE and DENS on **Benchmark 1**. For the sake of a fair comparison, we use the same maximum number of function evaluations MAX$_{\text{NE}}$ described in [8]. For the DENS, the parameters N_p, p_{ns}, F and CR are set to 100, 0.05, 0.5 and 0.9, respectively. The mutation strategy is $DE/rand/1/bin$. The algorithm is terminated when the number of function evaluations NE reaches to MAX$_{\text{NE}}$, which is listed in Table II.

The results of DENS and other four algorithms over 30 runs are given in Table II, where "Mean" indicates the mean function value. The results of SaDE, NSDE and SaNSDE are taken from Table II, III and IV in [8].

From the results in Table II, it can be seen that DENS outperforms the other four algorithms on all unimodal functions except for f_6; on this function, all algorithms obtain the same result. For the multimodal functions with high-dimension ($f_8 - f_{13}$), DENS outperforms the other four algorithms on f_9 and f_{10}, while SaNSDE surpasses other algorithms on f_{12} and f_{13}. For function f_{11}, DE, SaNSDE and DENS obtain the same performance, while SaDE and NSDE fail to solve it. SaDE, NSDE and SaNSDE successfully solve f_8, while DE and DENS fail. However, DENS outperforms DE on this function. For the multimodal functions with low-dimension ($f_{14} - f_{21}$), all the four algorithms almost obtain the same performance. Fig. 2 presents the convergence characteristics in terms of the mean best fitness value of DE and DENS on four functions. It can be seen that DENS converges faster than DE in the whole evolution except for f_8; on this function, DENS performs better than DE at the middle and last stages of the evolution.

C. *Results on* ***Benchmark 2***

Besides the **Benchmark 1**, a new set of benchmark functions provided by CEC 2005 special session was used [18]. It includes 25 functions, where functions $f_{cec1} - f_{cec5}$ are unimodal, and the rest 20 functions $f_{cec6} - f_{cec25}$ are multimodal. In this paper, we only use the first 10 functions $f_{cec1} - f_{cec10}$ for the **Benchmark 2**. The dimension of the problems is set to 10 and 30 in the experiments. Detailed descriptions of these functions can be found in [18].

1) *Results for* $D = 10$:

In this subsection, we present a comparative study of DMS-PSO [21], CMA-ES [22], [23], SaDE [5] and DENS on **Benchmark 2** with $D = 10$. To have a fair competition, we use the same evaluation criterion described in [18] for all algorithms. For the DENS, the parameters are same as before, N_p, p_{ns}, F and CR are set to 100, 0.05, 0.5 and 0.9, respectively. The mutation strategy is $DE/rand/1/bin$. All the algorithms are terminated when the number of function evaluations NE reaches to 100,000 (MAX$_{\text{NE}} = 10,000 * D$) or the function error value is less than $1e - 08$.

The average function error values of DENS and other three algorithms over 25 runs are given in Table III, where "Mean" indicates the mean function error value. The results of DMS-PSO, CMA-ES and SaDE are taken from Table XIII in [20].

From the results in Table III, it can be seen that both DENS and CMA-ES achieve the results less than the predefined error vale $1e-08$ for unimodal functions $f_{cec1} - f_{cec5}$, while DMS-PSO and SaDE fail on 2 and 3 functions, respectivley. For the rest five multimodal functions $f_{cec6} - f_{cec10}$, CMA-ES and DENS successfully solve two functions, while DMS-PSO and SaDE fail to solve all.

TABLE III
THE COMPARISON OF DENS WITH DMS-PSO, CMA-ES AND SADE ON BENCHMARK 1

Functions $D = 10$	DMS-PSO Mean	CMA-ES Mean	SaDE Mean	DENS Mean
f_{cec1}	**1e–09**	**1e–09**	**1e–09**	**1e–09**
f_{cec2}	**1e–09**	**1e–09**	**1e–09**	**1e–09**
f_{cec3}	**1e–09**	**1e–09**	1.67e–02	**1e–09**
f_{cec4}	1.89e–03	**1e–09**	1.42e–05	**1e–09**
f_{cec5}	1.14e–06	**1e–09**	1.2e–02	**1e–09**
f_{cec6}	6.89e–08	**1e–09**	1.20e–08	**1e–09**
f_{cec7}	4.59e–02	**1e–09**	2.00e–02	0.2649
f_{cec8}	**2.00e+01**	**2.00e+01**	**2.00e+01**	2.03e+01
f_{cec9}	3.62	**7.96e–02**	4.97	6.92
f_{cec10}	4.62	9.34e–01	4.89	**1e–09**

2) *Results for* $D = 30$:

In this subsection, we present a comparative study of SaDE, NSDE, SaNSDE and DENS on **Benchmark 2** with $D = 30$. To have a fair competition, we use the same maximum number of function evaluations (MAX$_{\text{NE}}$ $= 10,000 * D$) described in [8] for all algorithms. For the DENS, the parameters N_p, p_{ns}, F and CR are same as previous experiment.

TABLE IV
THE COMPARISON OF DENS WITH SADE, NSDE AND SANSDE ON BENCHMARK 2

Functions $D = 30$	SaDE Mean	NSDE Mean	SaNSDE Mean	DENS Mean
f_{cec1}	**0**	**0**	**0**	**0**
f_{cec2}	1.25e–13	4.11	**5.78e–14**	5.76e–07
f_{cec3}	1.77e+05	1.67e+06	**5.43e+04**	3.88e+05
f_{cec4}	1.89e+02	8.27e+01	1.22e–04	**1.01e–05**
f_{cec5}	1.00e+03	1.15e+03	**2.45e–01**	5.58
f_{cec6}	2.99e+01	2.89e+01	1.59e–01	**3.15e–02**
f_{cec7}	1.65e–02	1.12e–02	**8.57e–03**	**9.04e–03**
f_{cec8}	**2.09e+01**	**2.09e+01**	**2.09e+01**	**2.09e+01**
f_{cec9}	2.27e–15	1.99e–01	**0**	4.20e+01
f_{cec10}	5.15e+01	4.24e+01	4.21e+01	**0**

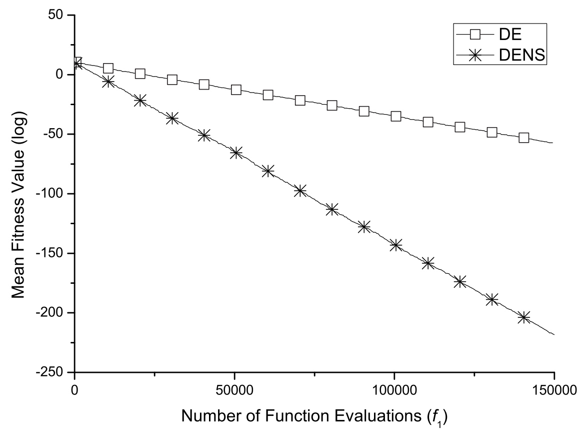
The average function error values of DENS and other three algorithms over 25 runs are given in Table IV, where "Mean" indicates the mean function error value. The results of SaDE, NSDE and SaNSDE are taken from Table V and VI in [8].

From the results in Table IV, it can be seen that DENS and SaNSDE outperform SaDE and NSDE on majority of test functions. Both SaNSDE and DENS almost achieve the same performance on functions f_{cec1}, f_{cec7} and f_{cec8}. DENS outperforms SaNSDE on functions f_{cec4},f_{cec6} and f_{cec10}, while SaNSDE achieves better results than DENS on f_{cec2}, f_{cec3}, f_{cec5} and f_{cec9}.

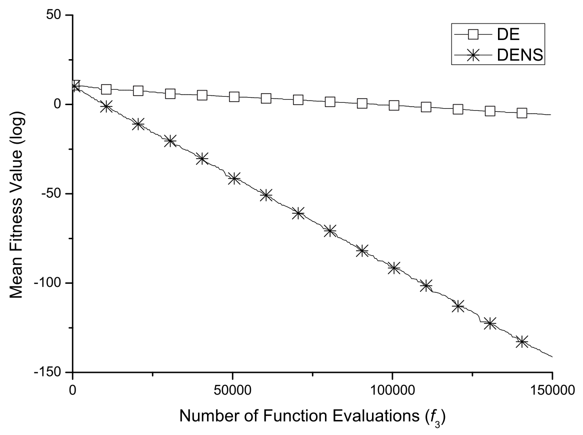
From the results of $D = 10$ and $D = 30$, the dimension greatly affects the performance of the algorithms. With the increasing of D, the problems become difficult to solve because of the increasing complexity, such as f_{cec3} and f_{cec5}. From the comparison of DENS with other algorithms on $D = 10$ and $D = 30$, DENS obtains similar performance with CMA-ES and SaNSDE on the majority of test functions. The results show that our approach is not a robust algorithm for all kinds of problems. However, the major advantage of the proposed algorithm is that the DENS is very simple, easy to implement, and obtaining promising performance on the majority of test functions.

V. CONCLUSIONS

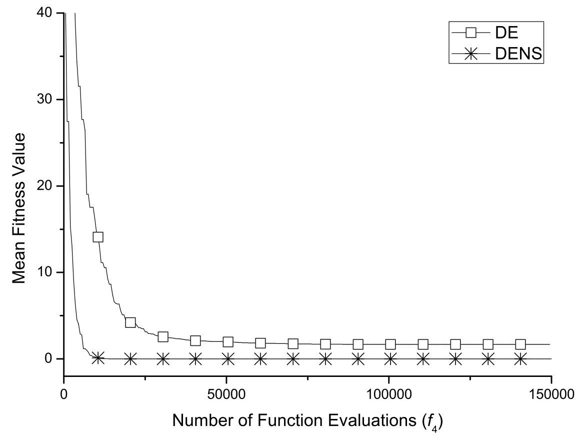
In this paper, we propose a new DE variants enhanced by neighborhood search, namely DENS, which differs from other previous works on neighborhood search based DEs, such as NSDE, SaNSDE and DEGL. The proposed approach modifies the basic neighborhood mutation schemes used in DEGL inspired by the mechanism of PSO. The DENS focuses on searching the neighbors by creating two trial vectors around the current vector. If one of the trial vectors is better than the current one, then replace the current vector with the better trial; otherwise keeps the current one unchangeable. In order to verify the performance of DENS, we provided a



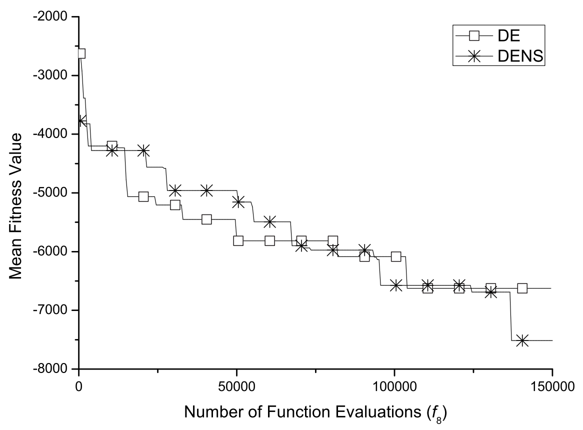
(a) f_1



(b) f_3



(c) f_4



(d) f_8

Fig. 2. Performance comparison between DE and DENS on functions f_1, f_3, f_4 and f_8.

comparative study of DENS and other six similar algorithms on two sets of benchmark functions. The results show that the DENS obtains promising performance on the majority of test functions.

Compared with other DE variants with neighborhood search, the concept of DENS is very simple and easy to implement, while SaNSDE is difficult to implement because of its complex steps in calculating the self-adaptive control parameters. Moreover, the modified neighborhood search strategies in DENS can be easily applied to other population-based algorithms.

We may combine the simple parameter strategy used in NSDE with DENS to obtain better performance. More experiments will be conducted in our future work, such as hybridization of DENS and other self-adaptive parameter mechanisms, and applying the proposed strategies to other evolutionary algorithms.

Acknowledgment

This work was supported by the Jiangxi Province Science & Technology Pillar Program (No.: 2009BHB16400), and the National Natural Science Foundation of China (No.s: 60871021, 60473037).

References

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

[2] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. Congr. Evol. Comput.*, 2004, vol. 2, pp. 1980–1987.

[3] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6 pp. 646–657, 2006.

[4] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol.9, no. 6, pp. 448–462, 2005.

[5] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. Congr. Evol. Comput.*, 2005, vol.2, pp. 1785–1791.

[6] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaption for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, 2009.

[7] Z. Yang, J. He, and X. Yao, "Making a difference to differential evolution," in *Advance in Metaheuristics for Hard Optimization*, 2008, pp. 397–414.

[8] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proc. Congr. Evol. Comput.*, 2008, pp. 1110–1116.

[9] J. Zhang and A. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5 pp. 945–958, 2009.

[10] S. Das, A. Abraham, U. Chakraborty and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13(3), pp. 526–553, 2009.

[11] S. Rahnamayan, H. R. Tizhoosh and M. M. A. Salama, "Opposition-based differential evolution algorithms," in *Proc. Congr. Evol. Comput.*, 2006, pp. 2010–2017.

[12] S. Rahnamayan, H. R. Tizhoosh and M. M. A. Salama, "Opposition-based differential evolution for optimization of noisy problems," in *Proc. Congr. Evol. Comput.*, 2006, pp. 1865–1872.

[13] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1 pp. 64–79, 2008.

[14] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, "Opposition versus Randomness in Soft Computing Techniques," *Elsevier Journal on Applied Soft Computing*, Volume 8, March 2008, pp. 906-918.

[15] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1 pp. 107–125, 2008.

[16] H. Wang, Z. Wu, S. Rahnamayan, and L. Kang "A Scalability Test for Accelerated DE Using Generalized Opposition-Based Learning," in *Proc. Intelligent System Design and Applications*, 2009, pp. 1090–1095.

[17] J. Kennedy and R. C. Eberhart, "Particle swarm optimization,' in *Proc. IEEE Int. Conf. Neural Networks*, 1995, pp. 1942–1948.

[18] 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, 2005.

[19] X. Yao, Y. Liu ,and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 82–102, 1999.

[20] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms behavior: a case study on the CEC2005 Special Session on Real Parameter Optimization," *J. Heuristics*, vol. 15, pp. 617–644, 2009.

[21] J.J. Liang, P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with local search," in *Proc. Congr. Evol. Comput.*, 2005, pp. 522–528.

[22] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proc. Congr. Evol. Comput.*, 2005, pp. 1769–1776.

[23] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," in *Proc. Congr. Evol. Comput.*, 2005, pp. 1777–1784.