Sequential DE Enhanced by Neighborhood Search for Large Scale Global Optimization

Hui Wang, Zhijian Wu, Shahryar Rahnamayan and Dazhi Jiang

***Abstract*— In this paper, the performance of a sequential Differential Evolution (DE) enhanced by neighborhood search (SDENS) is reported on the set of benchmark functions provided for the CEC2010 Special Session on Large Scale Global Optimization. The original DENS was proposed in our previous work, which differs from existing works which are utilizing the neighborhood search in DE, such as DE with neighborhood search (NSDE) and self-adaptive DE with neighborhood search (SaNSDE). In SDENS, 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. Additionally, sequential DE (DE with one-array) is used as a parent algorithm to accelerate the convergence speed in large scale search spaces. The simulation results for twenty benchmark functions with dimensionality of one thousand are reported.**

***Index Terms*— Differential evolution, neighborhood search, local search, large scale global optimization, high dimensional.**

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].

In this paper, a novel sequential DE algorithm enhanced by neighborhood search (SDENS) is proposed to improve the performance of DE. The DENS was introduced in our previous work [3] which presented two neighborhood search strategies to improve the quality of candidate solutions. In order to deal with large scale optimization problems, we employ a sequential DE (one-array DE) to accelerate the convergence speed. The performance of the algorithm is evaluated on the set of benchmark functions provided for the CEC2010 Special Session on Large Scale Global Optimization [4].

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).

Dazhi Jiang is with the Department of Computer Science, Shantou University, Shantou 515063 China (e-mail: jiangdazhi111007@tom.com).

The rest of the paper is organized as follows. In Section II, the classical DE algorithm is briefly reviewed. A short review of related works on large scale global optimization is presented in Section III. Section IV describes the proposed approach, SDENS. In Section V, the test benchmark functions, parameter settings besides a comprehensive set of scalability benchmarking are provided. Finally, the work is summarized and concluded in Section VI.

II. DIFFERENTIAL EVOLUTION

There are several variants of DE [1], where the most popular variant is indicated by *DE/rand/1/bin* which is called classical version. 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 behind the 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, for the two-array DE. For one-array DE (sequential DE), the selected vector is replaced in the same array. Fig. 1 and 2 present the schemes of two-array DE and one-array DE, respectively.

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

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

$$i \neq i_1 \neq i_2 \neq i_3,$$

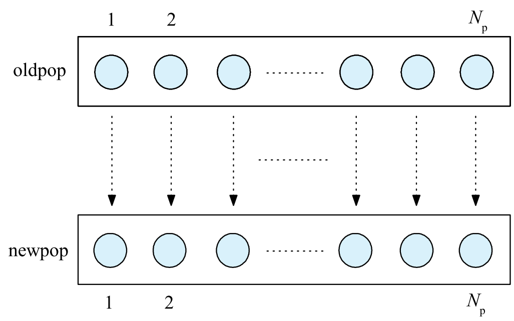


Fig. 1. The scheme of two-array DE.

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

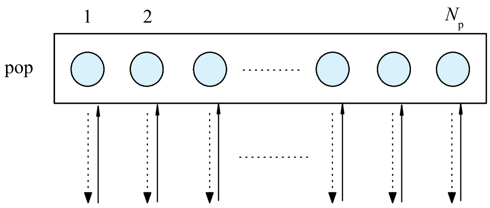


Fig. 2. The scheme of one-array DE (sequential DE).

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:

$$U_{i,G} = (U_{i,1,G}, U_{i,2,G}, \ldots, U_{i,D,G}), \tag{2}$$

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

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

$CR \in (0, 1)$ 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:

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

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. Related Works

Although classical evolutionary algorithms (EAs) have shown good optimization performance in solving some lower dimensional problems ($D < 100$), many of them suffers from the *curse of dimensionality*, which implies that their performance deteriorates quickly as the the dimensional size increases. The main reason is that in general the complexity of the problem increases with the size of its dimension. The majority of evolutionary algorithms lack the power of searching the optima solution when dimensionally increases. So more efficient search strategies are required to explore all the promising regions in a given time budget [5].

To improve the performance of population-based algorithms on large scale optimization problems, some interesting works have been proposed in the past two years. Yang *et al*. [6] proposed a multilevel cooperative co-evolution algorithm based on self-adaptive neighborhood search DE (SaNSDE) to solve large scale problems. Hsieh *et al*. [7] presented an efficient population utilization strategy for PSO (EPUS-PSO) to manage the population size. Brest *et al*. [8] introduced a population size reduction mechanism into self-adaptive DE, where the population size decreases during the evolutionary process. Tseng and Chen [9] presented multiple trajectory search (MTS) by using multiple agents to search the solution space concurrently. Zhao *et al*. [10] used dynamic multi-swarm PSO with local search (DMS-PSO) for large scale problems. Rahnamayan and Wang [11] presented a experimental study of opposition-based DE (ODE) [12] on large scale problems. The reported results show that ODE significantly improves the performance of standard DE. Wang and Li [13] proposed a univariate EDA (LSEDA-gl) by sampling under mixed Gaussian and lévy probability distribution. Rahnamayan and Wang [14] introduced an effective population initialization mechanism when dealing with large scale search spaces. Wang *et al*. [15] proposed an enhanced ODE based on generalized opposition-based learning (GODE) to solve scalable benchmark functions. Molina *et al*. [16] presented a memetic algorithm by employing MTS and local search chains to deal with large scale problems.

IV. Sequential DE Enhanced by Neighborhood Search Mechanism

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 sequential DE algorithm, called SDENS, to search the neighborhoods of individuals. The proposed approach differs from previous neighborhood search strategies in DE [17] or PSO [18]. Moreover, all other versions used two-array DE but SDENS uses one-array DE. Before introducing the SDENS, we need to give a brief review of other DE variants equipped by neighborhood search.

Yang *et al*. [19] 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 [20], [21] and NSDE, Yang *et al.* [22] 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 to calculate a better crossover rate CR. The presented experimental results show that SaNSDE outperforms SaDE and NSDE. As seen, these two DE variants with neighborhood search focus on the adaption of the control parameters.

We need to support a tradeoff between *exploration* and *exploitation* in most of EAs. The former indicates the global search ability and makes the algorithm explore every region of the feasible search space, while the latter means the local search ability and accelerates the algorithm converging to the near-optimal solutions. Most improvements on EAs try to seek a balance between these two factors which is suitable for different kinds of problems. The *DE/target-to-best/1* mutation strategy (described in Eq.5) promotes *exploitation* since all the individuals move to the same best position by the attraction of X_{best}, thereby results converging faster to that point [23]. 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 (premature convergence). To tackle this problem, Das *et al.* [23] proposed an enhanced DE algorithm (DEGL) by using an improved *DE/target-to-best/1* strategy which employs two mutation strategies: local neighborhood and global neighborhood.

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

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 $i \neq r_1 \neq r_2$.

Local Neighborhood Mutation–In the local model, each individual 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 global best point. The modified model is defined by

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

where the subscript n_best_i indicates the best individual 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 individuals $X_{n\_best_i,G}$, $X_{p,G}$ and $X_{q,G}$ are defined on a small neighborhood of $X_{i,G}$, and the searching behavior of each individual is almost independent. The information of individuals spread through the population regarding the best position of each neighborhood. Therefore, the attraction toward a specific points is weaker, which prevents the population from getting trapped into local minima [23].

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.

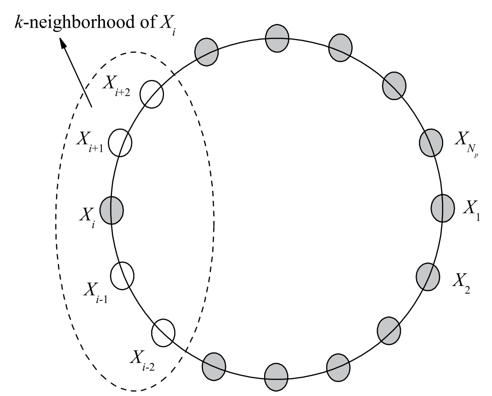


Fig. 3. The k-neighborhood in a ring topology, where $k = 2$.

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

where the subscript X_{best} indicates the best individual 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.

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

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. 3 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 based on 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

Algorithm 1: Sequential DE Enhanced by Neighborhood Search (SDENS).

Randomly initialize each individual in the population $P(G)$;
Calculate the fitness value of each $X_{i,G}$;
FEs $= N_p$;
Initialize $X_{pbest_i,G}$ and $X_{best,G}$;
while *FEs* $\leq$ *MAX_FEs* **do**
  for $i = 1$ *to* N_p **do**
    `/* Execute DE with hybrid crossover strategy */`
    Randomly select 3 vectors $X_{i_1,G}$, $X_{i_2,G}$ and $X_{i_3,G}$ from $P(G)$, where $i \neq i_1 \neq i_2 \neq i_3$;
    if $rand(0,1) \leq 0.5$ **then**
      Generate the trail vector $U_{i,G}$ using *rand/1/bin*;
    end
    else
      Generate the trail vector $U_{i,G}$ using *rand/1/exp*;
    end
    Calculate the fitness value of $U_{i,G}$;
    FEs $=$ FEs $+ 1$;
    if $f(U_{i,G}) < f(X_{i,G})$ **then**
      $X_{i,G} = U_{i,G}$
    end
    Update $X_{pbest_i,G}$ and $X_{best,G}$, if needed;
    `/* Conduct the neighborhood search */`
    if $rand(0,1) \leq p_{ns}$ **then**
      Create two trial vectors $L_{i,G}$ and $G_{i,G}$ according to Eq.11 and Eq.12, respectively;
      Calculate the fitness values of $L_{i,G}$ and $G_{i,G}$;
      FEs $=$ FEs $+ 2$;
      Select the fittest vectors from $\{X_{i,G}$, $L_{i,G}$ and $G_{i,G}\}$ as new $X_{i,G}$;
    end
    $X_{i,G+1} = X_{i,G}$;
  end
  $G = G + 1$;
end

$X_{n\_best_i,G}$, $X_{p,G}$ and $X_{q,G}$ are not the nearest neighbors to $X_{i,G}$ in the entire population.

In this paper, we propose another neighborhood search scheme which is inspired from the basic idea behind of DEGL [23] and also particle swarm optimization (PSO) [24]. 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 from 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 from 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 defined by

$$L_{i,G} = X_{i,G} + \alpha \cdot (X_{pbest_i,G} - X_{i,G}) + \beta \cdot (X_{p,G} - X_{q,G}), \quad (9)$$

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

$$L_{i,G} = (1-\alpha) \cdot X_{i,G} + \alpha \cdot X_{pbest_i,G} + \beta \cdot (X_{p,G} - X_{q,G}). \quad (10)$$

To simply the scaling factors $(1-\alpha)$, α and β in Eq.10, we use three correlated 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.

$$L_{i,G} = a_1 \cdot X_{i,G} + a_2 \cdot X_{pbest_i,G} + a_3 \cdot (X_{p,G} - X_{q,G}). \quad (11)$$

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

$$G_{i,G} = a_1 \cdot X_{i,G} + a_2 \cdot X_{best,G} + a_3 \cdot (X_{r_1,G} - X_{r_2,G}), \quad (12)$$

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 correlated 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, SDENS, 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 $X_{i,G}$.

Hybrid Crossover Strategy–According to suggestions of [25], DE with exponential crossover (*rand/1/exp*) shows better performance than binomial crossover (*rand/1/bin*) to solve high-dimensional problems. However, our empirical studies demonstrate that the exponential crossover is not suitable for all kinds of test functions. For some functions, the binomial crossover is more beneficial. To make a balance between these two crossover schemes, we use a hybrid crossover strategy as follows.

$$\begin{cases} rand/1/bin, & \text{if } rand(0,1) \leq 0.5 \\ rand/1/exp, & \text{otherwise} \end{cases}, \quad (13)$$

where $rand(0,1)$ is a random number within $[0,1]$.

The DENS has been proposed in our previous work [3] includes two operations, classical DE and neighborhood search, which are conducted in two different populations. In order to accelerate the convergence speed on large scale optimization, in this paper, we have utilized a sequential DENS, which executes classical DE and the neighborhood search in the same one population. The main steps of the SDENS 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, FEs is the number of function evaluations, and MAX_FEs is the maximum number of function evaluations.

TABLE I

RUNTIME ON THE TEST SUITE

System	Windows XP (SP3)
CPU	Intel (R) Core (TM)2 Duo CPU T6400 (2.00GHz)
RAM	2 G
Language	Java
Algorithm	SDENS
Runs/problem	25
MAX_FEs	3e+6
Dimension	1000
Runtime	78.6 hours

V. SIMULATION RESULTS

The proposed SDENS algorithm was tested on 20 benchmark functions provided by CEC2010 Special Session on Large Scale Global Optimization [4]. The parameter settings of SDENS are described as follows. The population size, N_p, is set to 50 based on empirical studies. The control parameters F and CR are set to 0.5 and 0.9, respectively [12]. The p_{ns} is set to 0.05 by the suggestions of our previous work [3]. The maximum number of functions evaluations MAX_FEs is set to 3e+6 for all test functions. The algorithm is conducted 25 runs for each test function, and the best, median, worst, mean and standard deviation of the error values are recorded.

The runtime of SDENS for the test suite are listed in Table I. For each test function, SDENS conducts 25 runs and the whole experiment on the 20 test functions cost about 97 hours.

Table II presents the results of SDENS on given 20 test functions. From the results, it can be seen that SDENS achieves good results only on four functions F_1, F_3 and F_6. For the rest of functions, especially for functions F_4, F_5, $F_7 - F_9$, F_{12}, F_{14}, F_{17} and $F19$, it could hardly find better solutions. The average converge curves on F_2, F_5, F_8, F_{10}, F_{13}, F_{15}, F_{18} and F_{20} are illustrated in Fig. 4.

VI. CONCLUSION REMARKS

In this paper, sequential DE enhanced by neighborhood search (SDENS) is proposed. The main idea of SDENS is to create two neighbors around the current individual by one local and one global neighborhood mutation operators. By simultaneously concerning the current individual and its two newly generated neighbors, we have more chance to find better solutions. Moreover, a one-array mechanism is used to accelerate the convergence speed. The performance of SDENS algorithm was evaluated on the set of benchmark functions provided by CEC2010 Special Session on Large Scale Global Optimization.

Compared with other DE variants with neighborhood search, the concept behind of SDENS 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 SDENS can be easily applied to other population-based algorithms.

In the experiments, the parameter settings of N_p and p_{ns} highly determine the performance of SDENS, and this paper only presents an empirical study. More investigations will be conducted to adjust these factors in the future work.

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] H. Wang, Z. Wu and S. Rahnamayan, "Differential evolution enhanced by neighborhood search," submitted to *Proc. Congr. Evol. Comput.*, 2010.

[4] K. Tang, X. Li, P. N. Suganthan, Z. Yang and T. Weise, "Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization," *Technical report*, Nature Inspired Computation and Applications Laboratory, USTC, China, 2010. http://nical.ustc.edu.cn/cec10ss.php.

[5] K. Tang, X. Yao, P. N. Suganthan C. Macnish, Y. Chen, C. Chen, Z. Yang, "Benchmark functions for the CEC'2008 special session and competition on high-dimensional real-parameter optimization," *Technical report*, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007. http://nical.ustc.edu.cn/cec08ss.php.

[6] Z. Yang, K. Tang, X. Yao, "Multilevel cooperative coevolution for large scale optimization, in *Proc. Congr. Evol. Comput.*, 2008, pp. 1663–1670.

[7] S. Hsieh, T. Sun, C. Liu, S. Tsai, "Solving large scale global optimization using improved particle swarm optimizer, in *Proc. Congr. Evol. Comput.*, 2008, pp. 1777–1784.

[8] J. Brest, A. Zamuda, B. Bošković, M. S. Maučec, V. Žumer, "High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction, in *Proc. Congr. Evol. Comput.*, 2008, pp. 2032–2039.

[9] L. Tseng, C. Chen, "Multiple trajectory search for large scale global optimization, in *Proc. Congr. Evol. Comput.*, 2008, pp. 3057–3064.

[10] S. Zhao, J. Liang, P. N. Suganthan, M. F. Tasgetiren, "Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization, in *Proc. Congr. Evol. Comput.*, 2008, pp. 3846–3853.

[11] S. Rahnamayan, G. Gary Wang, Solving large scale optimization problems by opposition-based differential evolution (ODE), World Scientific and Engineering Academy and Society, in *Transactions on Computers*, Volume 7, Issue 10, pp. 1792–1804, 2008.

[12] 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.

[13] Y. Wang, B. Li, A restart univariate estimation of distribution algorithm sampling under mixed Gaussian and Lévy probability distribution, in *Proc. Congr. Evol. Comput.*, 2008, pp. 3918–3925.

[14] S. Rahnamayan, G.G. Wang, Toward effective initialization for large-scale search spaces, World Scientific and Engineering Academy and Society, in *Transactions on Systems*, Volume 8, Issue 3, pp. 355-367, 2009.

[15] 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.

[16] D. Molina, M. Lozano and F. Herrera, "Memetic algorithm with local search chaining for continuous optimization problems: A scalability test," in *Proc. Intelligent System Design and Applications*, 2009, pp. 1068–1073.

[17] U. K. Chakraborty, S. Das and A. Konar, "Differential Evolution with Local Neighborhood," in *Proc. Congr. Evol. Comput.*, 2006, pp. 7395–7402.

[18] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proc. Congr. Evol. Comput.*, 1999, pp. 1958–1962.

TABLE II

THE RESULTS ACHIEVED BY SDENS ON THE TEST SUITE

FEs		F_1	F_2	F_3	F_4	F_5	F_6	F_7
$FEs = 1.2e5$	Best	3.93e+09	1.16e+04	1.99e+01	3.90e+13	3.14e+08	9.88e+05	3.07e+10
	Median	4.74e+09	1.19e+04	2.01e+01	4.60e+13	3.32e+08	2.03e+06	3.57e+10
	Worst	6.19e+09	1.20e+04	2.02e+01	7.90e+13	3.41e+08	2.39e+06	4.70e+10
	Mean	5.01e+09	1.19e+04	2.01e+01	5.10e+13	3.29e+08	1.84e+06	3.75e+10
	Std	9.18e+08	9.89e+01	1.17e–01	1.46e+13	1.04e+07	4.77e+05	5.46e+09
$FEs = 6e5$	Best	3.82e+06	7.00e+03	5.13e+00	8.47e+12	1.51e+08	1.38e+01	5.73e+09
	Median	4.59e+06	7.12e+03	6.27e+00	1.53e+13	1.83e+08	1.53e+01	7.73e+09
	Worst	1.95e+07	7.17e+03	6.76e+00	2.85e+13	2.12e+08	1.74e+01	1.36e+10
	Mean	7.87e+06	7.09e+03	6.12e+00	1.72e+13	1.81e+08	1.53e+01	9.28e+09
	Std	5.94e+06	6.76e+01	6.30e–01	6.68e+12	2.29e+07	1.18e+00	3.44e+09
$FEs = 3e6$	Best	1.75e–06	2.14e+03	1.23e–05	3.26e+12	7.66e+07	1.53e–04	6.36e+07
	Median	2.54e–06	2.17e+03	2.35e–05	3.72e+12	1.17e+08	1.76e–04	8.57e+07
	Worst	1.16e–05	2.39e+03	5.50e–05	8.99e+12	1.52e+08	2.57e–04	2.39e+08
	Mean	5.73e–06	2.21e+03	2.70e–05	5.11e+12	1.18e+08	2.02e–04	1.20e+08
	Std	4.46e–06	8.95e+01	1.54e–05	2.16e+12	2.88e+07	4.29e–05	6.56e+07
		F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
$FEs = 1.2e5$	Best	6.05e+08	1.13e+10	1.37e+04	2.27e+02	2.71e+06	1.70e+10	1.42e+10
	Median	6.23e+08	1.52e+10	1.38e+04	2.27e+02	2.83e+06	1.91e+10	1.73e+10
	Worst	1.20e+09	1.89e+10	1.42e+04	2.28e+02	3.29e+06	2.01e+10	2.31e+10
	Mean	7.71e+08	1.56e+10	1.39e+04	2.27e+02	2.95e+06	1.88e+10	1.84e+10
	Std	2.27e+08	2.77e+09	2.51e+02	3.49e–01	2.37e+05	1.07e+09	3.56e+09
$FEs = 6e5$	Best	4.64e+07	1.78e+09	1.02e+04	2.25e+02	1.25e+06	4.37e+05	3.91e+09
	Median	6.40e+07	2.13e+09	1.09e+04	2.26e+02	1.30e+06	6.67e+05	5.02e+09
	Worst	1.09e+08	2.88e+09	1.15e+04	2.26e+02	1.42e+06	7.64e+05	6.93e+09
	Mean	7.41e+07	2.23e+09	1.10e+04	2.26e+02	1.32e+06	6.43e+05	5.14e+09
	Std	2.73e+07	3.70e+08	4.59e+02	3.83e–01	5.98e+04	1.10e+05	9.89e+08
$FEs = 3e6$	Best	3.96e+07	4.77e+08	5.78e+03	2.20e+02	3.80e+05	1.16e+03	1.61e+09
	Median	4.09e+07	5.75e+08	7.03e+03	2.21e+02	3.95e+05	1.80e+03	1.86e+09
	Worst	9.35e+07	6.38e+08	7.37e+03	2.22e+02	4.97e+05	4.13e+03	2.30e+09
	Mean	5.12e+07	5.63e+08	6.87e+03	2.21e+02	4.13e+05	2.19e+03	1.88e+09
	Std	2.12e+07	5.78e+07	5.60e+02	5.09e–01	4.28e+04	1.03e+03	2.33e+08
		F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}	
$FEs = 1.2e5$	Best	1.36e+04	4.15e+02	3.84e+06	2.00e+11	1.19e+07	2.39e+11	
	Median	1.45e+04	4.15e+02	4.25e+06	2.09e+11	1.57e+07	2.62e+11	
	Worst	1.45e+04	4.15e+02	4.98e+06	2.35e+11	2.31e+07	2.82e+11	
	Mean	1.43e+04	4.15e+02	4.31e+06	2.11e+11	1.67e+07	2.61e+11	
	Std	3.72e+02	1.08e–01	4.04e+05	1.27e+10	3.71e+06	1.49e+10	
$FEs = 6e5$	Best	7.32e+03	4.13e+02	1.96e+06	1.65e+08	4.92e+06	1.36e+08	
	Median	1.18e+04	4.13e+02	2.02e+06	1.86e+08	5.39e+06	2.78e+08	
	Worst	1.26e+04	4.14e+02	2.29e+06	3.00e+08	6.18e+06	3.52e+08	
	Mean	1.03e+04	4.13e+02	2.07e+06	2.02e+08	5.41e+06	2.69e+08	
	Std	2.29e+03	3.49e–01	1.17e+05	5.02e+07	4.31e+05	7.57e+07	
$FEs = 3e6$	Best	7.14e+03	4.03e+02	8.78e+05	1.16e+04	7.57e+05	9.81e+02	
	Median	7.32e+03	4.09e+02	1.14e+06	3.32e+04	8.02e+05	9.83e+02	
	Worst	7.44e+03	4.10e+02	1.18e+06	4.51e+04	1.19e+06	1.02e+03	
	Mean	7.32e+03	4.08e+02	1.08e+06	3.08e+04	8.80e+05	9.90e+02	
	Std	9.63e+01	2.53e+00	1.11e+05	1.22e+04	1.59e+05	1.62e+01	

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

[20] 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.

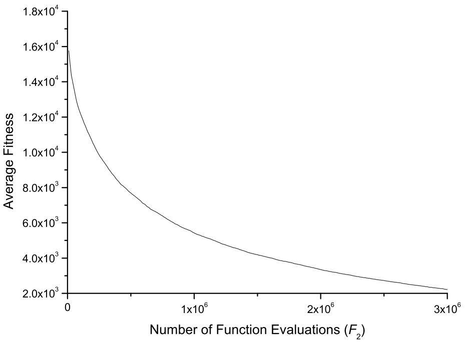
[21] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, pp. 398-417, 2009.

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

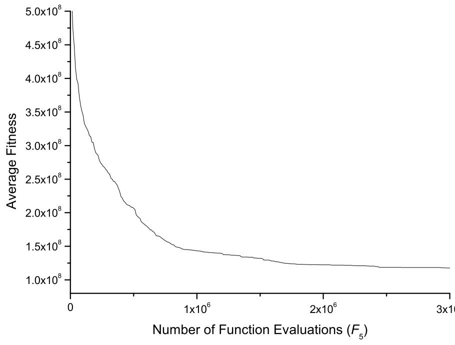
[23] S. Das, A. Abraham, U. Chakraborty and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13(3), pp. 526–553, 2009.

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

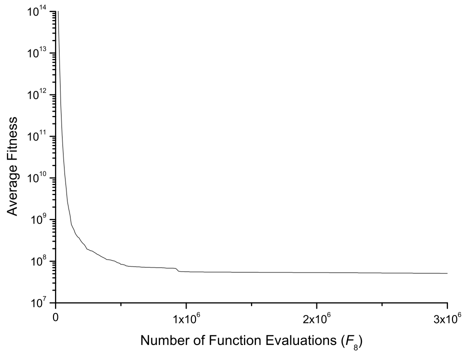
[25] F. Herrera, M. Lozano, D. Molina, "Components and Parameters of DE, Real-coded CHC, and G-CMAES," *Technical Report*, University of Granada, Spain, 2010.



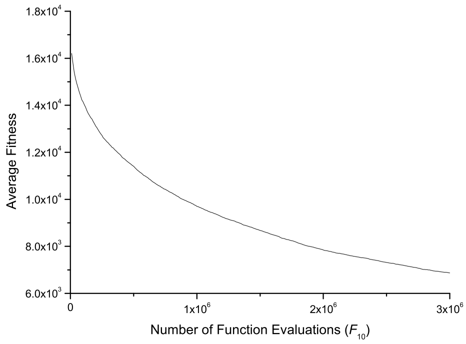
(a) F_2



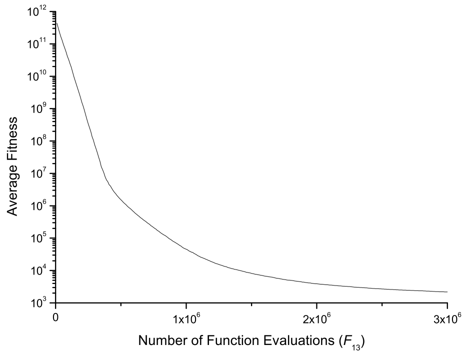
(b) F_5



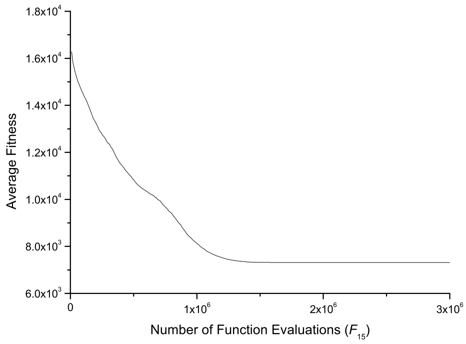
(c) F_8



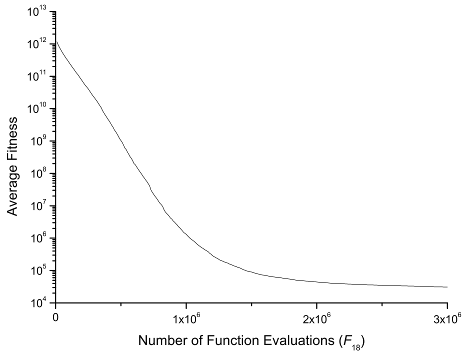
(d) F_{10}



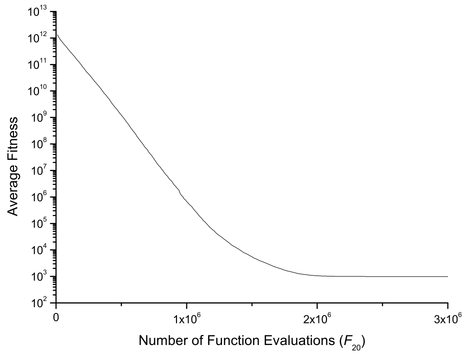
(e) F_{13}



(f) F_{15}



(g) F_{18}



(h) F_{20}

Fig. 4. The average convergence curves of SDENS on F_2, F_5, F_8, F_{10}, F_{13}, F_{15}, F_{18} and F_{20}.