Hindawi Publishing Corporation
Journal of Optimization
Volume 2016, Article ID 3260940, 14 pages
http://dx.doi.org/10.1155/2016/3260940

Research Article

Hybridization of Adaptive Differential Evolution with an Expensive Local Search Method

Rashida Adeeb Khanum,[1] Muhammad Asif Jan,[2] Nasser Mansoor Tairan,[3] and Wali Khan Mashwani[2]

[1]*Department of Mathematics, Jinnah College for Women, University of Peshawar, Khyber Pakhtunkhwa 25000, Pakistan*
[2]*Department of Mathematics, Kohat University of Science & Technology (KUST), Kohat, Khyber Pakhtunkhwa 26000, Pakistan*
[3]*College of Computer Science, King Khalid University, Abha 61321, Saudi Arabia*

Correspondence should be addressed to Muhammad Asif Jan; majan.math@gmail.com

Received 27 December 2015; Revised 9 June 2016; Accepted 14 June 2016

Academic Editor: Manlio Gaudioso

Differential evolution (DE) is an effective and efficient heuristic for global optimization problems. However, it faces difficulty in exploiting the local region around the approximate solution. To handle this issue, local search (LS) techniques could be hybridized with DE to improve its local search capability. In this work, we hybridize an updated version of DE, adaptive differential evolution with optional external archive (JADE) with an expensive LS method, Broydon-Fletcher-Goldfarb-Shano (BFGS) for solving continuous unconstrained global optimization problems. The new hybrid algorithm is denoted by DEELS. To validate the performance of DEELS, we carried out extensive experiments on well known test problems suits, CEC2005 and CEC2010. The experimental results, in terms of function error values, success rate, and some other statistics, are compared with some of the state-of-the-art algorithms, self-adaptive control parameters in differential evolution (jDE), sequential DE enhanced by neighborhood search for large-scale global optimization (SDENS), and differential ant-stigmergy algorithm (DASA). These comparisons reveal that DEELS outperforms jDE and SDENS except DASA on the majority of test instances.

1. Introduction

Optimization is concerned with finding best solution for an objective function. In general, an unconstrained optimization problem can be stated as follows: Find global optimum $\mathbf{x}^*$ of an objective function $f(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in R^n$ and n is the dimension of the problem.

Evolutionary algorithms (EAs) are inspired from Darwinian theory of evolution [1]. They are very efficient for finding global optimum of many real world problems, including problems from mathematics, engineering, economics, business, and medicines. EA family consists of a variety of stochastic algorithms, like Genetic Algorithms (GAs) [2], Particle Swarm Optimization (PSO) [3, 4], Evolutionary Strategies (ES) [5], and differential evolution algorithm (DE) [6, 7].

Among EAs, DE is the most recent algorithm and is efficient in solving many optimization problems. DE has many advantages. For example, it is simple to understand and implement, has a few control parameters, and is robust [8]. There is no doubt that DE is a remarkable optimizer for many optimization problems. But it has few limitations, like stagnation, premature convergence, and loss of population diversity [9, 10]. Being a global optimizer, DE suffers from searching the neighborhood of the approximate solution to the given problem. This makes room for hybridizing DE with other techniques to improve its poor exploitation (exploring the neighborhood of the approximate solutions). On the other hand, the role of LS methods is to stabilize the search especially in the environs of a local optimum. Thus, they can be combined with global search algorithms to enhance their local searching.

The main aim of this paper is to experiment with and validate the performance of our newly proposed hybrid algorithm, DEELS, which combines JADE [11, 12] and BFGS [13]. As a result, we want to see whether this hybridization will

improve the performance of JADE further. Contrary to our published preliminary work [14], this paper presents DEELS in full depth. It also comments on the performance of DEELS for large-scale global optimization problems with dimension 1000. Moreover, in contrast to our previous published comparison with JADE only [14], this time DEELS is compared with jDE [15], SDENS [16], and DASA [17] on problems from CEC2005 and CEC2010 test suits to further explore the capabilities of DEELS for handling small and large dimension problems.

The rest of this paper is organized as follows. Section 2 describes the basic DE, JADE, and the BFGS algorithms. Section 3 presents literature review. Section 4 presents proposed algorithm. Section 5 gives the experimental results, and finally Section 6 concludes this paper and discusses future research direction.

2. Some Relevant Existing Methods

As mentioned earlier, DEELS depends upon JADE and BFGS. Thus, this section presents the basic operators of DE, JADE, and BFGS.

2.1. Basic DE. Differential evolution (DE) [6, 7] is a recently developed bioinspired scheme for finding the global optimum $\mathbf{x}^*$ of an optimization problem. This section briefly reviews the DE algorithm. More details about it can be found in [18–22]. The working of DE can be described as follows.

2.1.1. Parent Selection. For each member $\mathbf{x}_i$, $i = 1, 2, \ldots, N_p$, of the current generation G, three other members, $\mathbf{x}_{r_1}, \mathbf{x}_{r_2}$, and $\mathbf{x}_{r_3}$, are randomly selected, where r_1, r_2, and r_3 are randomly chosen indices such that r_1, r_2, and $r_3 \in \{1, 2, \ldots, N_p\}$ and $i \neq r_1 \neq r_2 \neq r_3$. Thus, for each individual, $\mathbf{x}_i$, a mating pool of four individuals is formed in which an individual $\mathbf{x}_i$ breeds against three individuals and produces an offspring.

2.1.2. Reproduction. To generate an offspring, DE incorporates two genetic operators, mutation and crossover. They are detailed as follows:

(1) *Mutation*. After selection, mutation is applied to produce a mutant vector $\mathbf{v}_i$, by adding a scaled difference of the two already chosen vectors to the third chosen vector; that is,

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F\left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right), \tag{1}$$

where $F \in (0, 1)$ is the scaling factor.

(2) *Crossover*. After mutation, the parameters of the parent vector $\mathbf{x}_i$ and mutant vector $\mathbf{v}_i$ are mixed by a crossover operator and a trial member $\mathbf{u}_i$ is generated as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } \text{rand}_j(0, 1) \leq \text{CR}; \\ x_i, & \text{otherwise}, \end{cases} \tag{2}$$

where $j \in \{1, 2, \ldots, n\}$.

2.1.3. Survival Selection. At the end, the trial vector generated in (2) is compared with its parent vector on the basis of its objective function value. The best of the two will get a chance to become a member of the the new generation; that is,

$$\mathbf{x}_{i+1} = \begin{cases} \mathbf{u}_i, & \text{if } f(\mathbf{u}_i) \leq f(\mathbf{x}_i); \\ \mathbf{x}_i, & \text{otherwise.} \end{cases} \tag{3}$$

2.2. JADE. JADE [11] is an adaptive version of DE which modifies it in three aspects.

2.2.1. DE/Current/to-pbest Strategy. JADE utilized two mutation strategies: one with external archive and the other without it. These strategies can be expressed as follows [11]:

$$\mathbf{v}_i = \mathbf{x}_i + F_i\left(\mathbf{x}^p_{\text{best}} - \mathbf{x}_i\right) + F_i\left(\mathbf{x}_{r_1} - \widetilde{\mathbf{x}}_{r_2}\right), \tag{4}$$

$$\mathbf{v}_i = \mathbf{x}_i + F_i\left(\mathbf{x}^p_{\text{best}} - \mathbf{x}_i\right) + F_i\left(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}\right), \tag{5}$$

where $\mathbf{x}^p_{\text{best}}$ is a vector chosen randomly from the top p% individuals and $\mathbf{x}_i$, $\mathbf{x}_{r_1}$, and $\mathbf{x}_{r_2}$ are chosen from the current population P, while $\widetilde{\mathbf{x}}_{r_2}$ is chosen randomly from $P \cup A$, where A denotes the archive of JADE and p is a constant chosen as 0.5. In DEELS, we will utilize the strategy given in (4).

2.2.2. Control Parameters Adaptation. For each individual $\mathbf{x}_i$, control parameter F_i and the crossover probability CR_i are generated independently from Cauchy and normal distributions, respectively, as follows [11]:

$$F_i = \text{rand}\left(\mu F, 0.1\right), \tag{6}$$

$$\text{CR}_i = \text{rand}\left(\mu\text{CR}, 0.1\right). \tag{7}$$

These are then truncated to $(0, 1]$ and $[0, 1]$, respectively. Initially, both μF and μCR are set to 0.5. They are then updated at the end of each generation as follows:

$$\mu F = (1 - c)\,\mu F + c \cdot \text{mean}_L\left(S_F\right), \tag{8}$$

$$\mu\text{CR} = (1 - c)\,\mu\text{CR} + c \cdot \text{mean}_A\left(S_{\text{CR}}\right), \tag{9}$$

where mean_L denotes the Lehmer mean and mean_A denotes the arithmetic mean and S_F is the set of successful F_i's while S_{CR} is the set of successful CR_i's at generation G.

2.2.3. Optional External Archive. At each generation, the failed parents are sent to the archive. If the archive size exceeds N_p, some solutions are randomly deleted from it to keep its size equal to N_p. The archive inferior solutions play a roll in JADE's mutation strategy with archive. The archive not only provides information about direction but improves the diversity as well.

2.3. BFGS. The BFGS method, also known as the quasi Newton algorithm, employs the gradient and Hessian in finding a suitable search direction. BFGS is considered as a good LS method due to its efficiency. The detailed algorithm of BFGS is presented in Algorithm 1.

Input: *error*: desired accuracy;
γ: number of iterations.
$\mathbf{x}$: the starting vector.
$\mathbf{H}$: The Hessian matrix, initalize as identity matrix.
(1) $i = 0$.
(2) **while** $i < \gamma$ **do**
(3) Find the difference $\mathbf{d}_x = \mathbf{x}_{i+1} - \mathbf{x}_i$;
(4) Compute the difference of gradients $\mathbf{d}_g = \nabla f(\mathbf{x}_{i+1}) - \nabla f(\mathbf{x}_i)$;
(5) **if** $\mathbf{d}_x \neq 0$ and $\mathbf{d}_g \neq 0$ **then**
(6) $temp_1 = \mathbf{d}_g' \mathbf{H}_i \mathbf{d}_g$;
(7) $temp_2 = \mathbf{d}_x' \mathbf{d}_g$;
(8) Revise the Hessian matrix as:

$$\mathbf{H}_{i+1} = \mathbf{H}_i + \frac{(\mathbf{d}_x \mathbf{d}_x')(1 + (temp_1/temp_2))}{temp_2} - \frac{(\mathbf{H}_i \mathbf{d}_g \mathbf{d}_x' + \mathbf{d}_x \mathbf{d}_g' \mathbf{H}_i)}{temp_2}$$

(9) **end if**
(10) Compute the search direction $\mathbf{s}_i$ by using the current Hessian matrix $\mathbf{s}_i = -\mathbf{H}_i \nabla f(\mathbf{x}_i)$;
(11) Calculate α_i by golden section method [23];
(12) $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{s}_i$;
(13) **end while**
Output: $\mathbf{x}_{i+1}$ is the output of the algorithm.

Algorithm 1: Pseudocode of BFGS method [23].

3. Brief Review of Variants of DE and Hybridization of DE with Local Search Methods

To improve the performance of DE, many researchers devised modifications to the classic DE and proposed different variants. Some researchers modified the selection scheme [24], while others varied mutation and crossover operators [25]. Recently, in [26], orthogonal crossover was used instead of binomial and exponential crossover. Some have introduced new variants like opposition based DE (ODE) [27], centroid based initialization (ciJADE) [28], jDE [15], and genDE [8], while others introduced adaptation and self-adaptation of control parameters F and CR as in [29, 30], SaDE [31], JADE [11, 12], SHADE [32], and EWMA-DECrF [33]. Some introduced cooperative coevolution into DE for large-scale optimization [34]. A group of researchers applied it to discrete problems [35, 36], while others take advantage of its global search ability in continuous domains [26, 37–40].

In recent years, the hybridization of DE with LS methods has gained much attraction due to their individual merits. Many hybrid algorithms have shown significant performance improvement. Here, we review some of the methods in this category.

A new differential evolution algorithm with localization around the best point (DELB) is proposed in [41]. In DELB, the initial steps are the same as those in DE except that the mutation scale factor F is chosen from $[-1, -0.4] \cup [0.4, 1]$ randomly for each mutant vector. DELB also modifies the selection step by introducing reflection and contraction. The trial vector is compared with the current best and the parent vector. If the parent is worse than the trial vector, it is replaced by a new concentrated or reflected vector. In DELB, the trial vector can be replaced by its parent vector or reflected vector or contracted vector, while in classic DE only the trial vector replaces the parent.

Recently in [42], DE is hybridized with nonlinear simplex method. This method is known as NSDE. The authors of [42] applied nonlinear simplex method with uniform random numbers to initialize DE population. Initially, N_p individuals are generated uniformly and then next N_p are generated from these N_p points by application of Nelder-Mead Simplex (NMS). Now from $2N_p$ population, the fittest N_p are selected as DE's initial population and the rest of DE is unaltered in NSDE. Thus, NSDE modifies DE in the population step only. It has shown good performance in reducing function evaluations and CPU time.

In another experiment, Brest et al. [43] hybridized DE with Sequential Quadratic Programming (SQP), an efficient but expensive gradient-based LS method. Their hybrid applies the DE algorithm until function evaluations reach 30% of the maximum function evaluations. It then applies SQP for the first time to the best point thus obtained. Afterwards, SQP is applied after every 100 generations to the best solution of the current search. Expensive local search iteration number is set to $\lfloor \sqrt{\text{dimension}}/5 \rfloor$. In their hybrid, the population size keeps reducing and the process ends with minimum population size. DE provides the users with flexible offspring generation strategies [44]. Hence, hybridization of DE will continue to remain an active field of multidisciplinary research in the years to come.

Thus, we present a new algorithm, DEELS, which utilizes an expensive local search for refining the solutions. The details of DEELS are presented in the following section.

4. A New Hybrid Algorithm: DEELS

In this section, we present our new proposed algorithm, DEELS, which is the combination of two methods with contrasting features. First, we will discuss the main features of the algorithm. Then, we will describe it explicitly.

4.1. Main Idea. Though JADE, due to its adaptive parameter control strategy, performs better than classic DE on many optimization problems, however, its performance worsens with the increase in dimension. BFGS is a LS technique which has a strong self-correcting ability [45] in searching the optimal solution, but it is not globally as good as JADE. The important question is how to reconcile two different aspects to solve the minimization problem.

A very natural way would be to hybridize these two techniques, JADE and BFGS, together for solving unconstrained optimization problems. The issue is how to combine them in a way which is easy to understand and implement. Many hybrid approaches incorporate expensive methods to find the best solution. But, here, the new algorithm incorporates the robust and costly method not only for refining good solutions, but for locating them in the population during the search process.

DEELS begins with JADE and allows it to search for ξ generations. It then selects the q best individuals from this population and applies to them the expensive LS, that is, BFGS, for the first time. The objective of applying efficient search is to make them potential individuals to produce better offspring and lead the search in promising directions. These are then introduced into the population and the worse q solutions are removed from it.

The purpose of calling BFGS after ξ generations is to concentrate the population and add local search ability to the overall scheme and thus help it avoid getting trapped in the local optimal solutions. For these reasons, BFGS is invoked two more times in the evolution, with an interval of ξ generations. If function value is less than a threshold *error*, this means that it is in the neighborhood of the value to reach and this current best solution might lead the search to the desired optimal solution. Hence, it is desirable to apply the efficient LS by more than one iteration to this best solution. Thus, BFGS is applied by ς iterations when the best solution is in the vicinity of a local optimum. If the output solution of BFGS is the best known solution, then the algorithm stops; otherwise, it continues until the allowed maximum number of function evaluations is met.

In [43], the population size is reduced dynamically, while in our hybrid algorithm, we keep the population size fixed, since reducing the population size might result in losing population diversity, which is very important for DE. DEELS has got much inspiration from the state-of-the-art paper [46]. We apply expensive LS in combination with an EA (DE) instead of their inexpensive LS. In [46], both methods are LSs, while DEELS combines BFGS with JADE to investigate the effect of combing an EA with a LS method. In [46], a restart is also incorporated, while this is not necessary in DEELS.

4.2. Algorithmic Framework of DEELS. The details of DEELS are given in Algorithm 2. Here, we explain the different strategies used in DEELS.

4.2.1. Global Search. JADE improves the population of solutions by updating it from generation to generation with the help of genetic operators, mutation, and crossover. These operators help the search by producing promising solutions. JADE also possesses global search ability and thus adds it to DEELS. Moreover, JADE being a population based method can keep the diversity of the population and thus decreases the chances of DEELS getting trapped in local optima.

4.2.2. LS. The BFGS method has very strong self-correcting properties (when the right line search is used). If, at some iteration, the Hessian matrix contains bad curvature information, it has the ability to correct these inaccuracies by only few updates [45]. For this reason, BFGS generally performs very well, and once in the neighborhood of a minimizer it can attain superlinear convergence [45]. Though BFGS is efficient, it is a costly method, since it computes the gradient at the given point, which utilizes $2n$ function evaluations per gradient in DEELS. Further, it approximates the Hessian matrix $\mathbf{H}$, which is an $(n \times n)$ matrix of second-order partial derivatives [47], the computational cost of which is $O(n^2)$ per iteration [47]. BFGS needs $O(n^2)$ function evaluations per iteration [45]. Thus, the overall overhead of BFGS is also $O(n^2)$ per iteration.

The BFGS method plays two roles in DEELS; first, it is employed for generating promising solutions in the population after specified intervals of evolution. Secondly, it improves the quality of the best solution found so far by JADE and BFGS together.

Next, we explain what we mean by the terms concentration and refinement. As said earlier, the issue is to have an easy-to-understand and easy-to-implement search process. To achieve this, we need to rely on the fact that the problem is to distinguish between ordering points of which we have a lot and good ones (local optima) of which we have relatively few and the best points (global optima) of which we, potentially, may have only one or none.

Let us draw a diagram (see Figure 1) of the main process which is to rely on LS to do a course clustering (i.e., bring towards the basin of local optima of the majority of the good points in the population) and a refinement step in which hopefully the local optimum will be identified. It is clear that, initially, this process will be rather ineffective because of the sheer randomness of the population of solutions as shown in Figure 1(a); unless we are very lucky, it is unlikely to generate good points in the first population. But the important thing is that the process will become more and more effective as concentration takes its toll, on the population (see Figure 1(c)).

4.2.3. Updating the Population. Adding promising solutions to the population of DEELS and removing the worst points

(1) Inputs: Generate N_p uniform and random points, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_p}$ from the search space to form population P;
(2) q: the number of points selected for LS;
(3) γ: the number of iterations of LS for concentration;
(4) ς: the number of iterations of LS for refining solution;
(5) N_p: population size;
(6) FES: number of function evaluations;
(7) G: generation counter;
(8) ξ: interval between the LS calls;
(9) *error*: desired accuracy for LS method;
(10) $\mu\text{CR} = 0.5$, $\mu F = 0.5$;
(11) Evaluate the population;
(12) Set FES = N_p and $G = 0$;
(13) **while** FES < *MaxFeval* **do**
(14) Start the algorithm with JADE by using (4) for generating mutant vector, (2) for trial vector, (3) for best solution selection and (6) and (7) for adaptation of control parameters;
(15) Explore the population for ξ generations.
(16) Sort the objective values;
(17) Select q best points;
(18) **for** $i = 1$ to q **do**
(19) Apply γ iteration of BFGS to these q points;
(20) **if** $bestvalofBFGS < f(\mathbf{x}^*) + error_0$ **then**
(21) Break;
(22) **else if** $bestvalofBFGS < currentbestval$ **then**
(23) Update the population P by adding q new points to it such that its size becomes $N_p + q$;
(24) Sort the objective values;
(25) Delete the q worse individuals from P;
(26) **end if**
(27) **end for**
(28) Apply JADE to this new population until next ξ generations;
(29) **if** $f(\mathbf{x}) - f(\mathbf{x}^*) \leq error_0$ **then**
(30) Break;
(31) **else**
(32) $G = G + 1$;
(33) **end if**
(34) **end while**

Algorithm 2: Pseudocode of DEELS.

from it can improve the quality of offspring in the next generations. As if good parents can produce good offspring, worse parents also have the chance of producing worse solutions. Hence, their removal can have a good effect on the entire population. New potential solutions can also increase the convergence rate.

4.2.4. Stopping Condition. DEELS stops when one or both of the following conditions are met:

(1) The maximum number of function evaluations is reached.
(2) $|f(\mathbf{x}) - f(\mathbf{x}^*)| < error_0$, where $\mathbf{x}$ is the best individual found in a run and $\mathbf{x}^*$ is the known value to reach of the test instance.

The maximum number of function evaluations is set to $3 \times 10^{+06}$ for CEC2010 test instances with dimension 1000, while for 30-dimensional problems (CEC2005), these are chosen as $3 \times 10^{+05}$.

5. Comparison Studies

This section reports on two sets of experiments. In Experiment 1, DEELS is compared with jDE, while in Experiment 2,

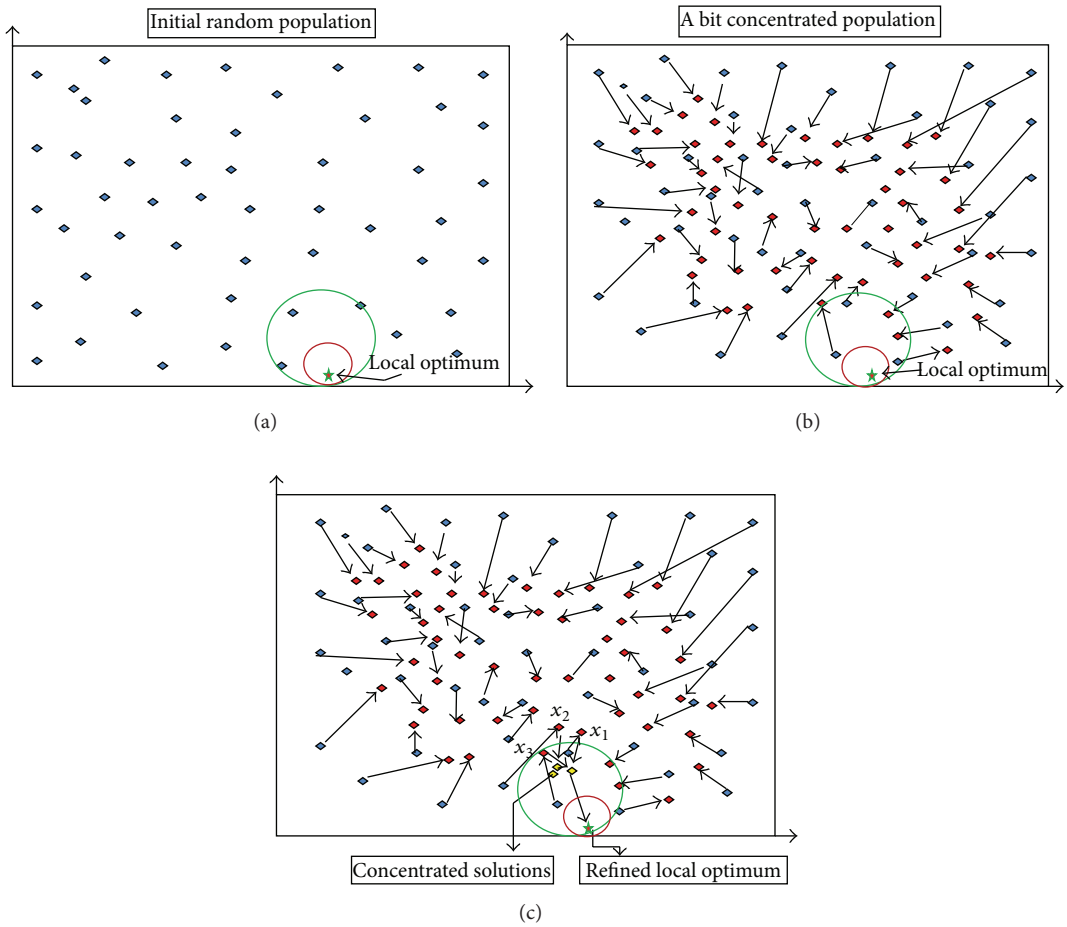

FIGURE 1: Concentration and refinement of solutions in a population.

DEELS is compared with SDENS and DASA. For comparison with SDENS and DASA, the experimental results for the best, median, mean, and standard deviation values are obtained from [17]. Moreover, all the experiments are conducted in MATLAB environment.

5.1. Experiment 1. In our preliminary results [14], DEELS was compared with JADE only, which is its internal optimization technique. However, here we compare DEELS with another state-of-the-art algorithm jDE [15], which is a self-adaptive DE variant for 30-dimensional problems.

5.1.1. Test Instances for Experiment 1. To study the performance of DEELS, we use CEC2005 test suit (see Table 1). This test suit was especially designed for single-objective unconstrained continuous optimization. Further, it was developed for low dimensions, for example, 30 and 50 dimensions. That is why we selected these instances for our experimental study. More details about these instances can be found in [48]. The instances of CEC2005 can be divided into the following:

(i) Unimodal test instances (F_1–F_5).

(ii) Multimodal test instances:

(1) Basic multimodal test instances (F_6–F_{12}),
(2) Expanded multimodal test instances (F_{13}-F_{14}).

(iii) Hybrid composition test instance (F_{15}).

The 15th test instance, F_{15} is designed by combining ten different benchmark functions, that is, two Rastrigin's functions, two Weirstrass's functions, two Griewank's functions, two Ackley's functions, and two Sphere functions. Its value to reach is 120.

5.2. Parameter Settings for Experiment 1. The population size N_p is set to 75, because N_p should be between $2n$ and $4n$ as suggested in [49]. Here, the problem dimension n is set to 30 for all the test instances in both jDE and DEELS. The other two parameters F and CR are initially set to 0.5, since this initial setting works well for all the test instances [11]. Later, the parameter values used in JADE are adopted. The number of elite solutions that undergo LS q is chosen as 3. The intensity of LS for concentration γ is set to 1 and the number of iterations of LS for refining the solution ς is set to 3.

Table 1: CEC2005 test instances.

Test instance	Test instances definition	$f(\mathbf{x}^*)$	Initialization range	Value to reach
F_1	Minimize $f_1 = \sum_{i=1}^{n} z_i^2 + f_{\text{bias}_1}$, where $\mathbf{z} = \mathbf{x} - \mathbf{o}$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where n is the dimension and $\mathbf{o} = (o_1, o_2, \ldots, o_n)$ is the shifted global optimum.	0	$[-100, 100]^n$	−450
F_2	Minimize $f_2 = \sum_{i=1}^{n} (\sum_{j=1}^{i} z_j)^2 + f_{\text{bias}_2}$, where $\mathbf{z} = \mathbf{x} - \mathbf{o}$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where n is the dimension and $\mathbf{o} = (o_1, o_2, \ldots, o_n)$ is the shifted global optimum.	0	$[-100, 100]^n$	−450
F_3	Minimize $f_3 = \sum_{i=1}^{n} (10^6)^{(i-1)/(n-1)} z_i^2 + f_{\text{bias}_3}$, where $\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where n is the dimension, $\mathbf{o} = (o_1, o_2, \ldots, o_n)$ is the shifted global optimum. $\mathbf{M}$ is the orthogonal matrix.	0	$[-100, 100]^n$	−450
F_4	Minimize $f_4 = (\sum_{i=1}^{n} (\sum_{j=1}^{i} z_j)^2) * (1 + 0.4\lvert N(0,1)\rvert) + f_{\text{bias}_4}$, where $\mathbf{z} = \mathbf{x} - \mathbf{o}$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where n is the dimension and $\mathbf{o} = (o_1, o_2, \ldots, o_n)$ is the shifted global optimum.	0	$[-100, 100]^n$	−450
F_5	Minimize $f_5 = \max\{\lvert \mathbf{A}_i \mathbf{x} - \mathbf{B}_i \rvert\} + f_{\text{bias}_5}$, where $i = 1, \ldots, n$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, n is the dimension, $\mathbf{A} = n \times n$ matrix, a_{ij} are random numbers $\in [-500, 500]$, $\det(\mathbf{A}) \neq 0$, $\mathbf{A}_i$ is the ith row of $\mathbf{A}$, $\mathbf{B}_i = \mathbf{A}_i * \mathbf{o}$, $\mathbf{o}$ is an $n * 1$ vector, and o_i are random numbers $\in [-100, 100]$.	0	$[-100, 100]^n$	−310
F_6	Minimize $f_6 = \sum_{i=1}^{n-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{\text{bias}_6}$, where $\mathbf{z} = \mathbf{x} - \mathbf{o}$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where n is the dimension and $\mathbf{o} = (o_1, o_2, \ldots, o_n)$ is the shifted global optimum.	0	$[-100, 100]^n$	390
F_7	Minimize $f_7 = \sum_{i=1}^{n} (z_i^2/4000) - \prod_{i=1}^{n} \cos(z_i/\sqrt{i}) + 1 + f_{\text{bias}_7}$, where $\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where n is the dimension and $\mathbf{o} = (o_1, o_2, \ldots, o_n)$ is the shifted global optimum. $\mathbf{M}$ is the linear transformation matrix, $\mathbf{M} = \mathbf{M}'(1 + 0.3\lvert N(0,1)\rvert)$.	0	$[0, 600]^n$	−180
F_8	Minimize $f_8 = -20\exp(-0.2\sqrt{(1/n)\sum_{i=1}^{n} z_i^2}) - \exp((1/n)\sum_{i=1}^{n} \cos(2\pi z_i)) + 20 + e + f_{\text{bias}_8}$, where $\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, n is the dimension, and $\mathbf{o} = (o_1, o_2, \ldots, o_n)$ is the shifted global optimum. $\mathbf{M}$ is the linear transformation matrix.	0	$[-32, 32]^n$	−140
F_9	Minimize $f_9 = \sum_{i=1}^{n} (z_i^2 - 10\cos(2\pi z_i) + 10) + f_{\text{bias}_9}$, where $\mathbf{z} = \mathbf{x} - \mathbf{o}$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where n is the dimension and $\mathbf{o} = (o_1, o_2, \ldots, o_n)$ is the shifted global optimum.	0	$[-5, 5]^n$	−330
F_{10}	Minimize $f_{10} = \sum_{i=1}^{n} (z_i^2 - 10\cos(2\pi z_i) + 10) + f_{\text{bias}_{10}}$, where $\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where n is the dimension and $\mathbf{o} = (o_1, o_2, \ldots, o_n)$ is the shifted global optimum. $\mathbf{M}$ is the linear transformation matrix.	0	$[-5, 5]^n$	−330
F_{11}	Minimize $f_{11} = \sum_{i=1}^{n} (\sum_{k=0}^{k\max} [a^k \cos(2\pi b^k (z_i + 0.5))]) - D\sum_{k=0}^{k\max} [a^k \cos(2\pi b^k \cdot 0.5)] + f_{\text{bias}_{11}}$, where $a = 0.5$, $b = 3$, $k\max = 20$, $\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}$, n is the dimension, $\mathbf{o} = (o_1, o_2, \ldots, o_n)$ is the shifted global optimum,	0	$[-0.5, 0.5]^n$	90

Table 1: Continued.

Test instance	Test instances definition	$f(\mathbf{x}^*)$	Initialization range	Value to reach
	$\mathbf{x} = (x_1, x_2, \ldots, x_n)$, and $\mathbf{M}$ is the linear transformation matrix.			
F_{12}	Minimize $f_{12} = \sum_{i=1}^{n} (\mathbf{A}_i - \mathbf{B}_i(\mathbf{x}))^2 + f_{\text{bias}_{12}}$, where $\mathbf{A}_i = \sum_{j=1}^{n} (a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j)$, $\mathbf{B}_i(x) = \sum_{j=1}^{n} (a_{ij} \sin x_j + b_{ij} \cos x_j)$, $i = 1, 2, \ldots, n$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, n is the dimension, $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n]$, and α_j are random integers $\in [-\pi, \pi]$. $\mathbf{M}$ is the linear transformation matrix.	0	$[-\pi, \pi]^n$	−460
F_{13}	Minimize $f_{13} = f_g(f_r(z_1 - z_2)) + f_g(f_r(z_2 - z_3)) + \cdots + f_g(f_r(z_{n-1} - z_n)) + f_g(f_r(z_n - z_1)) + f_{\text{bias}_{13}}$, Griewank's function is as follows: $f_g(x) = \sum_{i=1}^{n} (x_i^2/4000) - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$. Rosenbrock's function is as follows: $f_r(x) = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$, $\mathbf{z} = \mathbf{x} - \mathbf{o} + 1$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. n is the dimension. $\mathbf{o} = (o_1, o_2, \ldots, o_n)$ is the the shifted global optimum.	0	$[-3, 1]^n$	−130
F_{14}	Minimize $f_{14} = Ef(z_1 - z_2) + Ef(z_2 - z_3) + \cdots + Ef(z_{n-1} - z_n) + Ef(z_n - z_1) + f_{\text{bias}_{14}}$. f_{14} is the expansion of $f(x, y) = 0.5 + (\sin^2(\sqrt{x^2 + y^2}) - 0.5)/(1 + 0.001(x^2 + y^2))^2$, $\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M} + 1$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, and n is the dimension. $\mathbf{o} = (o_1, o_2, \ldots, o_n)$ is the shifted global optimum. $\mathbf{M}$ is the linear transformation matrix.	0	$[-100, 100]^n$	−300

Table 2: Experimental results of jDE and DEELS on 15 test instances of 30 variables with 3×10^5 FES. Mean error and std. dev. of the function error values obtained in 30 independent runs.

Test instance	jDE (mean error ± std. dev.)	DEELS (mean error ± std. dev.)	jDE SR	DEELS SR
F_1	$9.19E-09 \pm 7.42E-10$	$0.00E+00 \pm 0.00E+00^{\approx}$	100%	100%$^{\approx}$
F_2	$1.94E-08 \pm 1.89E-08$	$5.32E-08 \pm 9.48E-08^{\approx}$	**43.3%**	36.7%
F_3	$1.94E+05 \pm 9.95E+04$	**8.37E + 03 ± 5.94E + 03**	0	0
F_4	$3.90E-02 \pm 9.04E-02$	**9.73E − 09 ± 3.19E − 10**	0	**100%**
F_5	$8.21E+02 \pm 3.86E+02$	**3.30E − 05 ± 1.34E − 04**	0	**20.0%**
F_6	$6.08E+00 \pm 1.21E+01$	$6.15E+00 \pm 2.24E+01^{\approx}$	0	**86.7%**
F_7	$4.70E+03 \pm 1.93E-12$	**2.22E + 02 ± 2.51E + 01**	0	0
F_8	$2.09E+01 \pm 5.33E-02$	$2.09E+01 \pm 1.81E-01^{\approx}$	0	0
F_9	$8.97E-09 \pm 1.05E-09$	$5.21E-09 \pm 4.57E-09^{\approx}$	100%	100%$^{\approx}$
F_{10}	$4.89E+01 \pm 6.34E+00$	**2.19E + 01 ± 5.53E + 00**	0	0
F_{11}	$2.75E+01 \pm 1.48E+00$	**2.49E + 01 ± 1.75E + 00**	0	0
F_{12}	$4.30E+03 \pm 4.92E+03$	**4.17E + 03 ± 3.09E + 03**	0	0
F_{13}	$1.47E+00 \pm 1.49E-01$	**1.36E + 00 ± 8.91E − 02**	0	0
F_{14}	$1.30E+01 \pm 1.99E-01$	**1.23E + 01 ± 3.09E − 01**	0	0
F_{15}	$3.60E+02 \pm 9.32E+01$	**3.00E + 02 ± 1.31E + 02**	0	**3.3%**

The interval between the LS calls ξ is selected to be 300 generations which is equivalent to $300 \times N_p$ function evaluations.

5.3. Evaluation Metrics. Thirty independent runs were conducted for DEELS and jDE. The mean and standard deviation of the function error $|f(\mathbf{x}) - f(\mathbf{x}^*)|$ values are recorded for each run. We also record the success rate (SR) [31], for each test instance. A run is considered as successful if it achieves the desired accuracy within the maximum allowed function evaluations. The SR for a particular function is calculated as follows:

$$\text{SR} = 100 \times \frac{\text{Number of successful runs}}{\text{Number of total runs}}. \quad (10)$$

5.4. Comparison of DEELS with jDE. The experimental results for function error values and SR of jDE and DEELS are presented in Table 2. The convergence graphs of both algorithms are obtained by plotting the number of function evaluations against the objective function values. DEELS outperforms jDE on 10 out of 15 test instances, while on the remaining 5 test instances the performance of both algorithms is comparable. In the following, we comment on the DEEL's behavior in each category of test instances.

5.4.1. Unimodal Test Instances (F_1–F_5). As can be observed from Table 2, DEELS performed well for three out of five test instances, F_3–F_5 in terms of function error values. For the remaining two instances, F_1 and F_2, both algorithms are considered to be comparable.

Considering SR, here again DEELS performed well for two unimodal test instances, F_4 and F_5. jDE only showed a higher SR in the case of F_2. Overall, on unimodal test instances, DEELS is better than jDE, which can be observed in the last column of Table 2.

5.4.2. Multimodal Test Instances (F_6–F_{14}). In the case of multimodal test instances, DEELS performed very well on six test instances, F_7 and F_{10}–F_{14} in achieving a good solution (see Table 2). The graphs presented in Figure 2 for these multimodal test instances also prove that the yellow curve (jDE) is above the green curve (DEELS). This means that DEELSs obtained solutions are smaller than those obtained by jDE. This proves that DEELS outperforms jDE. Both algorithms showed equal performance on the rest (i.e., F_6, F_8, and F_9) of multimodal test instances: ≈ symbol in Table 2 shows this fact.

For F_9, both algorithms attained the 100% accuracy level as given in Table 2. For the remaining multimodal test instances, neither of the algorithms could reach the desired accuracy in any run except for F_6, on which DEELS obtained 86.7% SR over zero SR of jDE. Thus, one can conclude here again that DEELS is better than jDE in the case of multimodal test instances.

5.4.3. A Hybrid Composition Test Instance (F_{15}). This test instance, being the combination of other test functions, is a challenging test function. Hence, it is not an easy task to find its global optimum or attain 100% SR for it. DEELS is successful in getting a better local optimum for it than jDE

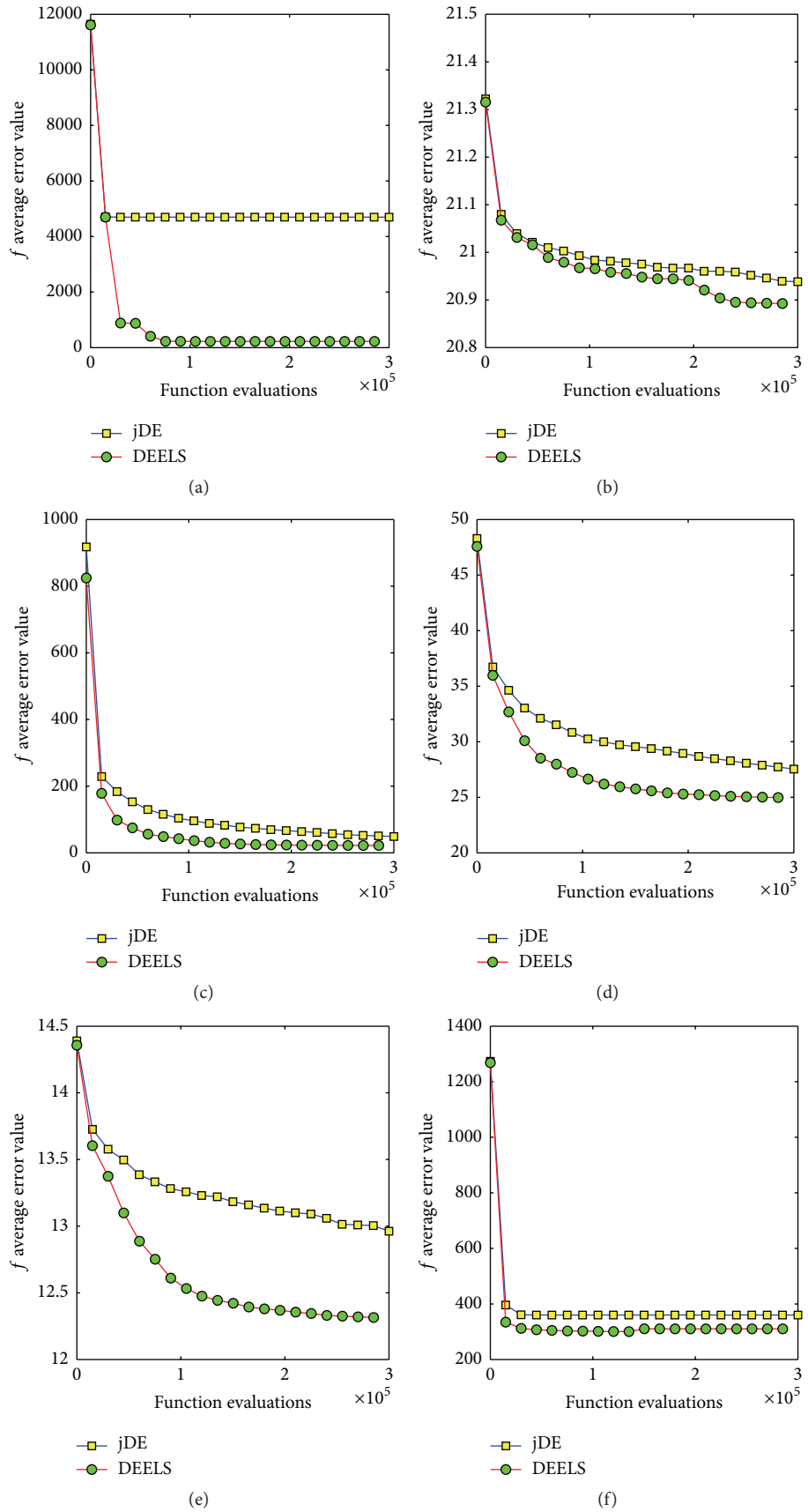

FIGURE 2: Convergence graphs of jDE and DEELS for six representative test functions at $n = 30$, with population size = 75. (a) F_7, (b) F_8, (c) F_{10}, (d) F_{11}, (e) F_{14}, and (f) F_{15}.

(please see the convergence graphs of Figure 2). DEELS also obtained a 3.3% SR for this test instance against 0 SR of jDE. This good performance of DEELS may be due to the fact that DEELS benefits from global search and LS, while jDE is only a global search method and so may not be good in exploiting better solutions.

In general, it is interesting to note that jDE, though equivalent to DEELS on five out of 15 test instances, could not get a better function error value than DEELS for any test instance.

5.5. Experiment 2. In this section, we compare DEELS first with SDENS [50] and then with DASA [17] on the CEC2010 test instances with problem dimension 1000.

5.5.1. Test Instances for Experiment 2. We further investigate the behavior of DEELS on ten new and complex test instances with problem dimension $n = 1000$, used in CEC2010 Special Session and Competition on Large-Scale Global Optimization [51]. The test instances used in our experiments are the first ten test instances of CEC2010, which can be divided into two categories as follows:

(i) Unimodal test instances (F_1, F_4, F_7 and F_9).

(ii) Multimodal test instances (F_2, F_3, F_5, F_6, F_8 and F_{10}).

5.5.2. Parameter Setting for Experiment 2. The parameters settings are kept the same as demanded in the original paper [51] for CEC2010 instances. For this experiment, the population size $N_p = 50$ is chosen and the problem dimension n is set to 1000. The maximum function evaluations are chosen as $3 \times 10^{+06}$. The value to reach is set to 10^{-2}. Twenty-five independent runs of DEELS have been performed for all test instances.

5.6. Comparison with SDENS. The best, median, mean, and standard deviation of function error values obtained in 25 runs of DEELS are presented in Table 3.

As can be seen from Table 3, overall DEELS performed well as compared with SDENS in reaching the best solution for seven out of ten test instances, F_2, F_4, F_5, F_7, to F_{10}. Surely, this better performance is due to the additional exploitation abilities of DEELS. For the remaining three test instances, F_1, F_3, and F_6, SDENS dominated the best solutions of DEELS. F_1 and F_3 are separable functions, while F_6 is a single-group nonseparable multimodal function. The mean value obtained by DEELS on F_6 in Table 3 is substantially larger than that of SDENS. Therefore, it may be reasonable to think that the failure of DEELS is due to the BFGS, which may get trapped at a local optimum.

It is interesting to note from Table 3 that, based on median and mean values, DEELS found consistently better median and mean of the average error values than SDENS for the seven out of ten test instances, F_2, F_4, F_5, and F_7–F_{10}. However, for the remaining three test instances, F_1, F_3, and F_6, SDENS maintained its dominance over DEELS. This poor performance of DEELS might be due to one of the abovementioned reasons.

Both algorithms, DEELS and SDENS, achieved 50% success based on standard deviation values as illustrated in Tables 3. That is, for five test instances, F_4, F_5, and F_7 to F_9, DEELS performed well in terms of standard deviation values, while for the other five test instances, F_1 to F_3, F_6, and F_{10}, SDENS outperforms DEELS.

Overall, DEELS performance was better than SDENS on best, median, and mean values. However, in case of standard deviation values, the performance of both algorithms is 50%.

5.7. Comparison with DASA. Table 3 presents the best, median, mean, and standard deviation values for DASA, which are obtained from [17]. This table shows that DEELS is superior to DASA on four test instances, F_4, F_6, F_8, and F_{10}, in terms of best function error values. These test instances are mainly m-group nonseparable except F_{10}, which is $n/2m$-group nonseparable; all are multimodal except F_4. For the remaining six test instances, DEELS performed poorly against DASA in achieving the best function error values. Please note that, among these six functions, F_1, F_2, and F_3 are separable. Table 3 shows that DEELS outperforms DASA in finding good median and mean of the function error values for the five test instances, F_4, F_5, F_6, F_8, and F_{10}, while it is inferior to DASA on the other five test instances, F_1 to F_3, F_7, and F_9; here, F_7 is m-group nonseparable and F_9 is $n/2m$-group nonseparable.

The worse performance of DEELS against DASA can be seen only in case of standard deviation values, where DEELS is superior only in three test instances, F_4, F_5, and F_8, while DASA surpasses DEELS on seven test instances, F_1 to F_3, F_6, F_7, F_9, and F_{10}.

Thus, one can conclude that based on the median and mean function error values DASA and DEELS have similar performance on nonseparable test functions except standard deviation and the minimum objective values where DASA is better than DEELS. The latter failed totally on separable test functions, F_1 to F_3. It also remained poor on two nonseparable test functions, F_7 and F_9.

6. Conclusion

In this paper, we described DEELS, a new hybrid algorithm that combines two well known algorithms, JADE and BFGS, to keep a balance between exploration and exploitation. DEELS showed efficient performance on majority of the tested test instances against jDE and SDENS except DASA. Based on the experimental results, it can be concluded that LS method can improve the local tuning of the solutions provided that these are hybridized at a proper gap with the global optimizer; otherwise, it can cause early termination of the algorithm and results in premature convergence. It is also observed that DEELS fails on separable functions.

Table 3: Experimental results of SDENS, DASA, and DEELS on 10 test instances of 1000 variables with $3 \cdot 10^{+06}$ FES. Best, median, mean, and std. dev. of the function error values obtained over 25 runs.

Test instance	Best			Median			Mean			Std. dev.		
	SDENS	DASA	DEELS	SDENS	DASA	DEELS	SDENS	DASA	DEELS	SDENS	DASA	DEELS
F_1	$1.75E-06^{+}$	$5.59E-23^{+}$	$4.58E+05$	$2.54E-06^{+}$	$5.42E-22^{+}$	$1.50E+06$	$5.73E-06^{+}$	$1.52E-21^{+}$	$3.11E+06$	$4.46E-06^{+}$	$2.33E-21^{+}$	$3.68E+06$
F_2	$2.14E+03^{+}$	$3.98E+00^{+}$	$4.82E+02$	$2.17E+03^{-}$	$7.96E+00^{+}$	$9.80E+02$	$2.21E+03^{-}$	$8.48E+00^{+}$	$9.28E+02$	$8.95E+01^{+}$	$2.52E+00^{+}$	$2.77E+02$
F_3	$1.23E-05^{+}$	$5.54E-11^{+}$	$9.89E-01$	$2.35E-05^{+}$	$7.37E-11^{+}$	$1.16E+00$	$2.70E-05^{+}$	$7.20E-11^{+}$	$1.20E+00$	$1.54E-05^{+}$	$8.27E-12^{+}$	$1.62E-01$
F_4	$3.26E+12^{-}$	$2.26E+11^{-}$	$7.27E+10$	$3.72E+12^{-}$	$4.94E+11^{-}$	$1.16E+11$	$5.11E+12^{-}$	$5.05E+11^{-}$	$1.19E+11$	$2.16E+12^{-}$	$2.22E+11^{-}$	$3.62E+10$
F_5	$7.66E+07^{-}$	$4.41E+08^{-}$	$3.52E+07$	$1.17E+08^{-}$	$6.36E+08^{-}$	$6.98E+07$	$1.18E+08^{-}$	$6.20E+08^{-}$	$6.86E+07$	$2.88E+07^{-}$	$7.87E+07^{-}$	$1.47E+07$
F_6	$1.53E-04^{+}$	$1.96E+07^{-}$	$1.97E+01$	$1.76E-04^{+}$	$1.97E+07^{-}$	$1.98E+01$	$2.02E-04^{+}$	$1.97E+07^{-}$	$7.03E+04$	$4.29E-05^{+}$	$4.45E+04^{+}$	$2.43E+05$
F_7	$6.36E+07^{-}$	$2.57E+00^{+}$	$5.98E+05$	$8.57E+07^{-}$	$7.18E+00^{+}$	$9.45E+05$	$1.20E+08^{-}$	$7.78E+00^{+}$	$9.87E+05$	$6.56E+07^{-}$	$3.10E+00^{+}$	$2.27E+05$
F_8	$3.96E+07^{-}$	$2.84E+03^{+}$	$1.22E+05$	$4.09E+07^{-}$	$1.21E+06^{-}$	$4.15E+05$	$5.12E+07^{-}$	$4.98E+07^{-}$	$4.05E+06$	$2.12E+07^{-}$	$8.95E+07^{-}$	$7.62E+06$
F_9	$4.77E+08^{-}$	$2.83E+07^{+}$	$3.83E+07$	$5.75E+08^{-}$	$3.58E+07^{+}$	$4.92E+07$	$5.63E+08^{-}$	$3.60E+07^{+}$	$4.97E+07$	$5.78E+07^{-}$	$4.78E+06^{+}$	$5.85E+06$
F_{10}	$5.78E+03^{-}$	$6.78E+03^{-}$	$4.00E+03$	$7.03E+03^{-}$	$7.33E+03^{-}$	$4.38E+03$	$6.87E+03^{-}$	$7.29E+03^{-}$	$4.79E+03$	$5.60E+02^{+}$	$2.69E+02^{+}$	$1.09E+03$
$-$	6	4		7	5		7	5		5	3	
$+$	4	6		3	5		3	5		5	7	
$\approx$												

"$-$," "$+$," and "$\approx$" denote that the performance of the SDENS and DASA algorithms is worse than, better than, and similar to that of DEELS, respectively.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1679–1696, 2011.

[2] S. Y. Yuen and C. K. Chow, "A genetic algorithm that adaptively mutates and never revisits," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 454–472, 2009.

[3] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micromachine and Human Science1995.*, pp. 39–43, Nagoya, Japan, 1995.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, December 1995.

[5] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Natural Computing Series, Springer, Berlin, Germany, 2003.

[6] R. Storn, *Differential Evolution (DE) Research-Trends and Open Questions*, vol. SCI 143, Springer, Berlin, Germany, 2008.

[7] R. Storn and K. V. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[8] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.

[9] Z.-F. Hao, G.-H. Guo, and H. Huang, "A particle swarm optimization algorithm with differential evolution," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, pp. 1031–1035, IEEE, Hong Kong, August 2007.

[10] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[11] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[12] C. Zhang and L. Gao, "An effective improvement of JADE for real-parameter optimization," in *Proceedings of the 6th International Conference on Advanced Computational Intelligence (ICACI '13)*, pp. 58–63, Hangzhou, China, October 2013.

[13] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, New York, NY, USA, 2nd edition, 1987.

[14] R. A. Khanum and M. A. Jan, "Hybridization of adaptive differential evolution with BFGS," in *Research and Development in Intelligent Systems XXIX: Incorporating Applications and Innovations in Intelligent Systems XX Proceedings of AI-2012, The Thirty-second SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 441–446, Springer, Berlin, Germany, 2012.

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

[16] H. Wang, Z. Wu, S. Rahnamayan, and D. Jiang, "Sequential DE enhanced by neighborhood search for large scale global optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1–7, IEEE, Barcelona, Spain, July 2010.

[17] P. Korosec, K. Tashkova, and J. Silc, "The differential ant-stigmergy algorithm for large scale global optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '10)*, pp. 1–8, IEEE, Barcelona, Spain, July 2010.

[18] R. Storn and K. Price, "Home page of differential evolution," Tech. Rep., 2003, http://www1.icsi.berkeley.edu/~storn/code.html.

[19] S. Das and P. N. Suganthan, "Tutorial: differential evolution, foundations, prospectives and applications," in *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI '11)*, pp. 1–59, Paris, France, April 2011.

[20] P. N. Suganthan and Swagatam, "Tutorial: differential evolution," in *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI '11)*, pp. 1–76, Paris, France, April 2011.

[21] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 61–106, 2010.

[22] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[23] P. Venkataraman, *Applied Optimization with Matlab Programming*, John Wiley & Sons, New York, NY, USA, 2002.

[24] V. V. D. Melo and A. C. Botazzo Delbem, "Investigating Smart Sampling as a population initialization method for differential evolution in continuous problems," *Information Sciences*, vol. 193, pp. 36–53, 2012.

[25] D. Zaharie, "A comparitive analysis of crossover varients in differential evolution," in *Proceedings of the International Multiconference on Computer Science and Information Technology*, Wisła, Poland, October 2007.

[26] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[27] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

[28] R. A. Khanum and M. A. Jan, "Centroid-based Initialized JADE for global optimization," in *Proceedings of the 3rd Computer Science and Electronic Engineering Conference (CEEC '11)*, pp. 115–120, IEEE, Colchester, UK, July 2011.

[29] J. Brest, A. Zamuda, B. Boskovic, S. Greiner, and V. Zumer, *Advances in Differential Evolution*, vol. SCI143 of *An Analysis of the Control Parameters' Adaptation in Differential Evolution*, 2008.

[30] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 1110–1116, IEEE Press, June 2008.

[31] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[32] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proceedings of the IEEE*

Congress on Evolutionary Computation (CEC '13), pp. 71–78, Cancún, Mexico, June 2013.

[33] J. Aalto and J. Lampinen, "A mutation and crossover adaptation mechanism for differential evolution algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 451–458, Beijing, China, July 2014.

[34] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Journal of Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.

[35] C.-S. Deng, B.-Y. Zhao, A.-Y. Deng, and C.-Y. Liang, "Hybrid-coding binary differential evolution algorithm with application to 0-1 knapsack problems," in *Proceedings of the International Conference on Computer Science and Software Engineering (CSSE '08)*, pp. 317–320, Wuhan, China, December 2008.

[36] C. S. Deng, B. Y. Zhao, and C. Y. Liang, "Hybrid binary differential evolution algorithm for 0-1 knapsack problem," *Computer Engineering and Design*, vol. 31, no. 8, pp. 1795–1798, 2010.

[37] C. Segura, C. A. C. Coello, E. Segredo, and C. León, "An analysis of the automatic adaptation of the crossover rate in differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 459–466, IEEE, July 2014.

[38] A. K. Qin, K. Tang, H. Pan, and S. Xia, "Self-adaptive differential evolution with local search chains for real-parameter single-objective optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 467–474, IEEE, Beijing, China, July 2014.

[39] F. Wei, Y. Wang, and T. Zong, "Variable grouping based differential evolution using an auxiliary function for large scale global optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 1293–1298, IEEE, Beijing, China, July 2014.

[40] R. A. Khanum, N. Tairan, M. A. Jan, W. K. Mashwani, and A. Salhi, "Reflected adaptive differential evolution with two external archives for large-scale global optimization," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, pp. 675–683, 2016.

[41] P. Kaelo and M. M. Ali, "A numerical study of some modified differential evolution algorithms," *European Journal of Operational Research*, vol. 169, no. 3, pp. 1176–1184, 2006.

[42] M. Ali, M. Pant, and A. Abraham, "Simplex differential evolution," *Acta Polytechnica Hungarica*, vol. 6, no. 5, pp. 95–115, 2009.

[43] J. Brest, A. Zamuda, B. Bošković, S. Greiner, M. S. Maučec, and V. Žumer, "Self-adaptive differential evolution with SQP local search," in *Proceedings of the 3rd International Conference on Bioinspired Optimization Methods and their Applications (BIOMA '08)*, pp. 59–69, Ljubljana, Slovenia, October 2008.

[44] S. Das, S. S. Mullick, and P. Suganthan, "Recent advances in differential evolution—an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[45] A. Skajaa, *Limited memory bfgs for nonsmooth optimization [M.S. thesis]*, 2010.

[46] F. J. Hickernell and Y. Yuan, "A simple multistart algorithm for global optimization," *OR Transactions*, vol. 1, no. 2, pp. 1–12, 1997.

[47] R. B. Schnabel, "Concurrent function evaluations in local and global optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 64, no. 1–3, pp. 537–552, 1987.

[48] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, vol. 2, pp. 1785–1791, September 2005.

[49] J. Ronkkonen, V. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 506–513, Edinburgh, Scotland, September 2005.

[50] H. Wang, Z. Wu, S. Rahnamayan, and D. Jiang, "Sequential DE enhanced by neighborhood search for large scale global optimization," in *Proceedings of the 6th IEEE World Congress on Computational Intelligence (WCCI '10)*, pp. 1–7, Barcelona, Spain, July 2010.

[51] K. Tang, P. N. Xiodongo, Z. Yang, and T. Weise, "Benchmark functions for the CEC2010 special session and competition on large scale global optimization," Tech. Rep., Nature Inspired Computation and Application Laboratory (NICAL), University of Science and Technology of China, 2010.