Complex & Intelligent Systems (2019) 5:25–40
https://doi.org/10.1007/s40747-018-0086-8

ORIGINAL ARTICLE

LSHADE-SPA memetic framework for solving large-scale optimization problems

Anas A. Hadi[1] · Ali W. Mohamed[2] · Kamal M. Jambi[1]

Received: 26 June 2018 / Accepted: 16 November 2018 / Published online: 4 December 2018

Abstract
During the last decade, large-scale global optimization has been one of the active research fields. Optimization algorithms are affected by the curse of dimensionality associated with this kind of complex problems. To solve this problem, a new memetic framework for solving large-scale global optimization problems is proposed in this paper. In the proposed framework, success history-based differential evolution with linear population size reduction and semi-parameter adaptation (LSHADE-SPA) is used for global exploration, while a modified version of multiple trajectory search is used for local exploitation. The framework introduced in this paper is further enhanced by the concept of divide and conquer, where the dimensions are randomly divided into groups, and each group is solved separately. The proposed framework is evaluated using IEEE CEC2010 and the IEEE CEC2013 benchmarks designed for large-scale global optimization. The comparison results between our framework and other state-of-the-art algorithms indicate that our proposed framework is competitive in solving large-scale global optimization problems.

Keywords Evolutionary computation · Global optimization · Large-scale optimization · Differential evolution of LSHADE-SPA · Memetic framework

Introduction

Optimization has been considered one of the main growing research fields during the last decade. Efficiently finding the maximum or minimum value of a function has a great impact on many real-world problems. Traditional (calculus-based) optimization approaches are the direct solution that can solve this problem. However, objective functions in many fields and real-world problems could face many issues that make them difficult—if not impossible—to be optimized using traditional optimization approaches. On the other hand, metaheuristic optimization approaches are an appropriate alternative to find approximate solutions to such problems.

Scientific and industrial real-world problems are growing in complexity, some of them need to solve up to thousands of variables [1]. With the increase in the dimensions, the complexity of these problems sometimes grows exponentially which is known as the curse of dimensionality [2]. Additionally, the huge increase in dimensions usually changes the properties of the search. A small-scale unimodal function may change to a multi-modal function when the number of dimensions increases. Moreover, the evaluation cost in LSGO is usually expensive which may affect the optimization process. Finally, the non-separability feature is considered a serious challenge for LSGO, especially if the concept of divide and conquer is used to handle such problems [3]. Optimization of such kind of problems is considered a challenging task.

Motivated by these challenges, many efficient, effective, and robust metaheuristic algorithms for LSGO problems with high-quality solution and low-computational cost have been proposed.

The rest of the paper is organized as follows: "Related work" covers the background of LSGO and reviews related

✉ Anas A. Hadi
anas1401@gmail.com

Ali W. Mohamed
aliwagdy@gmail.com

Kamal M. Jambi
kjambi@kau.edu.sa

[1] Faculty of Computing and Information Technology, King Abdulaziz University, P. O. Box 80200, Jeddah 21589, Saudi Arabia

[2] Operations Research Department, Institute of Statistical Studies and Research, Cairo University, Giza 12613, Egypt

works. "LSHADE semi-parameter adaptation memetic framework" will describe our proposed memetic framework (MLSHADE-SPA). In "Experimental results", experimental analysis of MLSHADE-SPA will be evaluated using CEC2010 and CEC2013, and the performance of MLSHADE-SPA will be compared with state-of-the-art LSGO algorithms. "Parametric analysis" introduces a parametric analysis regarding the impact of each component of our framework. Finally, a conclusion and summary will be given in "Conclusion".

Related work

During the last 10 years, IEEE Congress on Evolutionary Computation (IEEE CEC) has organized many large-scale global optimization (LSGO) special sessions and competitions. Due to its significance, LSGO has attracted much attention from the researchers all over the world. Generally, we can identify two main research directions for LSGO metaheuristic algorithms.

The first direction is the decomposition of the original problem into smaller problems. This is done mainly using cooperative coevolution (CC) algorithm. CC investigates decomposition methods that are able to identify groups of interacting variables. The performance of CC is affected by the correct choice of the decomposition method. To perform variable decomposition effectively, CC requires prior knowledge about the structure of the problems.

The second direction for LSGO metaheuristics is a hybridization of multiple methods. Using more than one metaheuristic algorithm in a collaborative way usually increases the performance of the algorithm. A clear success trend in this direction was achieved by the hybridization of population-based metaheuristic with local search metaheuristic. The first endorses the exploration, while the latter endorses the exploitations. The rest of this section will further discuss recent algorithms that are related to these two directions.

Multiple trajectory search (MTS) [4] was the winner of the first known LSGO competition CEC2008. MTS initializes a small population using simulated orthogonal arrays (SOA). Then, three local search methods adaptively participate to improve the best three agents; MTS-LS1 evaluates each variable independently, while MTS-LS2 and MTS-LS3 evaluate a group of variables at the same time.

Self-adaptive differential evolution with multi-trajectory search (SaDE-MMTS) [5] proposed a hybridization framework between JADE [6] and a modified version of MTS-LS1. Binomial, exponential, and no crossover strategies were adapted for SaDE-MMTS. In MMTS, the used search range *SR* is adaptively determined every time the MMTS is applied. The average distances between current population members are calculated and scaled using one of five predetermined linearly reducing factors (LRF). If a better solution is found, *SR* is further reduced.

MA-SW-Chains [7] was introduced in 2010. It is considered as an extension of MA-CMA-Chains. But due to scalability issues of CMA [8], CMA was replaced with Solis Wets (SW) [9] which is a more scalable LS algorithm. MA-SW-Chains is based on the concept of LS chain. It adjusts the LS intensity to act more intensely in the most promising areas. MA-SW-Chains was the winner of CEC2010 [10].

Ensemble Optimization Evolutionary Algorithm (EOEA) [11] was the second-ranked algorithm in CEC2010. In EOEA, optimization process has two stages, global shrinking, and local exploration. EDA based on mixed Gaussian and Cauchy models (MUEDA) [12] was used in EOEA to shrink the searching scope to the promising area as quickly as possible. The second stage objective is to explore the limited area to find better solutions. In this stage, a CC-based algorithm using SaDE [13], GA [14], and cooperative GA and DE [15] was used. The third rank in CEC2010 was for differential ant-stigmergy algorithm (DASA) [16]. It tries to solve LSGO by transforming the real-parameter optimization problem into a graph-search problem.

CEC2010 benchmark was also used in CEC2012 competition. Improved multiple offspring sampling (MOS) [17] was the winner of CEC2012. MOS combines Solis and Wets [9] and MTS-LS1 [4] as two local searches. These two algorithms are executed in sequence, and the amount of fitness evaluations is assigned adaptively to each algorithm according to its performance.

Self-adaptive differential evolution algorithm (jDElsgo) was proposed in [18]. Crossover and scaling factor control parameters were adapted during the optimization process. A population size reduction mechanism was also included in jDElsgo. Later on, jDElsgo with a small and varying population size (jDEsps) [19] was introduced as an improved version of jDElsgo. It varies the population size by starting with small size. Then, the population size is increased. After that, the population size reduction is used to reduce population size. JDElsgo was ranked second in CEC2012.

The third rank in CEC2012 was for cooperative coevolution evolutionary algorithm with global search (CCGS). It is considered an extension of EOEA [20]. In CCGS, CC-based EA is used to perform the exploration stage, while MUEDA is used to perform the exploitation stage.

To enhance the performance of CC framework on non-separable problems, cooperative coevolution with delta grouping (DECC-DML) was proposed in [21]. Delta grouping as a new decomposition strategy was proposed where the averaged difference in a certain variable across the entire population was measured and used for identifying the interacting variables.

In [22], differential evolution with landscape modality detection and a diversity archive (LMDEa) was proposed. LMDEa uses a small population size and large archive to control the diversity of the search. On the other hand, scaling factor F is controlled using landscape modality.

In [23], sequential differential evolution (DE) enhanced by neighborhood search (SDENS) was introduced. SDENS depends on a local and global neighborhood to create new individuals. DE is used as a population-based optimizer to accelerate the convergence speed. Two crossover operators, binomial an exponential, are used in SDENS.

A new benchmark was proposed for CEC2013 [24]. DECC-G [27] was the reference algorithm in CEC2013. It incorporates DE and CC framework with a group-based problem decomposition strategy. The winner of CEC2013 competition was modified MOS [25]. Solis and Wets [9], as well as MTS-LSI-Reduced as a modified version of MTS-LSl [4], were used as local search algorithms. On the other hand, GA [14] was included as a global exploration algorithm. To control the exploitative behavior of the algorithm, MTS-LSI-Reduced tries to identify expected better dimensions for the next generation.

The second-ranked algorithm was smoothing and auxiliary function-based cooperative coevolution for global optimization (SACC) [26]. In SACC, a parallel search is done first using CC. After that, the local solutions worse than the better ones are eliminated using a smoothing function.

In CEC2014, a new decomposition method based on high-dimensional model representation (HDMR) was proposed in [28]. It tries to identify the subcomponents before applying the optimization process. CEC2010 benchmark was used to evaluate the performance of this algorithm.

A hybrid adaptive evolutionary differential evolution (HACC-D) [29] was also proposed in CEC2014. JADE [10] and SaNSDE [30] were used as CC subcomponent optimization algorithms. CEC2010 benchmark was also used to evaluate the performance of HACC-D. Neither of the two algorithms, HDMR and HACC-D, outperformed the existing best results from previous competitions [31].

Variable grouping-based differential evolution (VGDE) [32] was also proposed in CEC2014. Variables interaction is detected and grouped using variable grouping strategy. To find better solutions, VGDE proposed an auxiliary function. The performance of VGDE was evaluated using CEC2013 and the results were competitive.

CEC2013 was the used benchmark in CEC2015. Iterative hybridization of DE with local search (IHDELS) was proposed in [33]. The best solution is shared between population-based and two LS-based methods in an iterative way. L-BFGSB [34] and MTS-LS-1 were used as LS methods, while DE was used as a population-based method. If the best solution cannot be improved, LS will randomly select another solution. IHDELS results were comparatively compared with MOS, the winner of CEC2013.

In CEC2016, CC with dependency identification grouping (DISCC) [35] was proposed. DISCC tries to find the most suitable arrangement for the variable using a dependency identification grouping mechanism. CEC2010 was used to evaluate the performance of DISCC. CBCC [36] was another CC proposed in CEC2016. CBCC is a contribution-based CC that allocates the computational resources to the components based on their contributions. The performance of CBCC was evaluated using only partially separable functions of the CEC2013. In the same year, Coral Reefs Optimization (CRO) was extended using different substrate layers and local search (CRO-LS) to solve LSGO [37].

Self-evaluation evolution (SEE) was proposed in [38], where the objective function is divided into sub-problems. Then, each sub-problem is assigned to an EA optimizer. During the optimization process, the optimizer and the search operators are trained to correctly evaluate the partial solutions.

In [39], CCFR-I and CCFR-IDG2 were proposed. SaNSDE [30] was used as the optimizer subcomponent. CCFR-I tries to allocate the computational resources among the subpopulations based on how they contribute to the global best improvement. CCFR-IDG2 is a variant of CCFR-I which tries to group the variables with a very high accuracy.

Recently, a multi-modal optimization based on CC (MMO-CC) was proposed in [40], MMO-CC searches for multiple optima and uses them as informative representatives to be exchanged among subcomponents.

In 2017, an enhanced adaptive differential evolution (EADE) [41] was introduced. A new mutation rule was proposed in EADE, where the vectors are chosen in a directed way. The best vector is chosen from the best 10% vectors, the worst vector is chosen from the worst 10% vectors, and the middle vector is chosen from the range between them. Crossover rate (CR) was adapted by a gradual change of the CR values according to the past experience. In the same year, ANDE as another adaptive DE with novel triangular mutation strategy was proposed [42]. This mutation selects three vectors randomly and sorts them from best to worst. Then, the convex combination is defined using them.

Finally, LSHADE with semi-parameter adaptation (SPA) hybrid with CMA-ES (LSHADE-SPACMA) was proposed in [43]. The concept of SPA is to enhance the adaptation of the scaling factor F and crossover rate Cr by changing one parameter at a time. During the first half of the search, the adaptation process is concentrated on Cr value, while F will be generated randomly. During the second half, Cr values are gradually frozen to the adapted values, while the adaption process is concentrated on F values. A modified version of CMA-ES was integrated with LSHADE-SPACMA framework, where crossover operation was applied to CMA-

ES to improve the exploration capability of the algorithm. LSHADE-SPACMA was evaluated using CEC2017 benchmark, which is considered as a moderate size benchmark. To solve LSGO problems, and due to scalability issues of CMA, LSHADE-SPACMA needs to be enhanced.

A comprehensive review and analysis regarding state-of-the-art evolutionary algorithms participating using the latest CEC benchmarks can be found in [31].

The main objective of this paper is to design a memetic framework (MLSHADE-SPA) which solves LSGO problems effectively. MLSHADE-SPA is a hybridization framework between population-based algorithms and local search. LSHADE-SPA, EADE, and ANDE are used as population-based algorithms for global exploration, while a modified version of MTS (MMTS) is used as a local search algorithm for local exploitation.

Furthermore, the concept of divide and conquer is used to enhance the performance of the framework. This procedure is done without any prior assumptions about the structure of the optimized problems, where the dimensions are randomly divided into groups, and each group is solved separately.

MLSHADE-SPA framework will be evaluated and compared with other state-of-the-art algorithms using CEC2010 and the CEC2013 benchmarks designed for LSGO. MLSHADE-SPA is compared with 26 recent algorithms that belong to different EAs classes: 17 of them are compared using CEC2010, and 9 are compared using CEC2013. To the best of our knowledge, this is the first study that uses all these different types of algorithms to carry out evaluation and comparisons.

LSHADE semi-parameter adaptation memetic framework

In this section, the details of LSHADE-SPA memetic (MLSHADE-SPA) framework will be described. LSHADE-SPA, EADE, ANDE, and MMTS will be covered first. After that, we will discuss the proposed MLSHADE-SPA framework in details.

LSHADE semi-parameter adaptation

To establish a starting point for the optimization process, an initial population P^0 must be created. Typically, each jth component ($j = 1, 2, \ldots, D$) of the ith individuals ($i = 1, 2, \ldots, \text{NP}$) in the P^0 is obtained as the following:

$$x_{j,i}^{0} = x_{j,L} + \text{rand}(0,1)\left(x_{j,U} - x_{j,L}\right) \tag{1}$$

where rand (0,1) returns a uniformly distributed random number in [0, 1].

At generation G, for each target vector x_i^G, a mutant vector v_i^G is generated according to current-to-pbest/1 mutation strategy which was proposed in JADE [6]:

$$v_i^G = x_i^G + F_i^G \cdot \left(x_{\text{pbest}}^G - x_i^G\right) + F_i^G \cdot \left(x_{r1}^G - x_{r2}^G\right) \tag{2}$$

P value in pbest is used to balance exploitation and exploration by controlling the greediness of the mutation strategy. r_1 is a random index selected from the population. r_2 is another random index selected from the concatenation of the population and an external archive. This external archive holds parent vectors which successfully produced better vectors. x_{pbest}^G is the best individual vector with the best fitness value in the population at generation G. The scale factor F_i^G is a positive control parameter for scaling the difference vector.

In the crossover, the target vector is mixed with the mutated vector, using the following scheme, to yield the trial vector u_i^G:

$$u_{i,j}^G = \begin{cases} v_{i,j}^G \; if\left(\text{rand}_{i,j} \le Cr\, OR\, j = j_{\text{rand}}\right) \\ x_{i,j}^G \; \text{otherwise} \end{cases} \tag{3}$$

where $rand_{j,i}\; i \in \{1, N\}\; and\; j \in \{1, D\}$ are uniformly distributed random numbers in [0,1], $Cr \in [0,1]$ called the crossover rate that controls how many components are inherited from the mutant vector, j_{rand} is a uniformly distributed random integer in $[1, D]$ that makes sure at least one component of trial vector is inherited from the mutant vector.

DE adapts a greedy selection strategy. u_i^G is set to x_i^{G+1} if and only if the trial vector u_i^G yields as good as or a better fitness function value than x_i^G. Otherwise, the old vector x_i^G is reserved. The selection scheme is as follows (for a minimization problem):

$$x_i^{G+1} = \begin{cases} u_i^G,\, if\left(f\left(u_i^G\right) \le f\left(x_i^G\right)\right) \\ x_i^G,\, \text{otherwise} \end{cases} \tag{4}$$

To improve the performance of LSHADE-SPA, linear population size reduction (LPSR) was used. In LPSR, the population size will be decreased according to a linear function. The linear function in LSHADE-SPA was:

$$N_{G+1} = \text{round}\left[\left(\frac{N^{\min} - N^{init}}{\text{MAX}_{\text{NFE}}}\right) * \text{NFE} + N^{\text{init}}\right] \tag{5}$$

where NFE is the current number of fitness evaluations, MAX_{NFE} is the maximum number of fitness evaluations, N^{init} is the initial population size, and $N^{\min} = 4$ which is the minimum number of individuals that DE can work with.

To perform semi-parameter adaptation (SPA) for F and Cr, the adaptation process is composed of two parts. The idea is to activate the "one-factor-at-a-time" policy. Thus, during the first half of the search, the adaptation will be concentrated on one parameter Cr, while F parameter will be generated using uniform distribution randomly within a specific limit. During the second half, the adaptation will be concentrated on F, while Cr parameter will be gradually frozen to the adapted values.

Enhanced adaptive differential evolution

To balance the global exploration ability and the local exploitation, EADE mutation strategy was proposed in [41] as follows:

$$v_i^G = x_r^G + F_1 \cdot \left(x_{P\_\text{best}}^G - x_r^G\right) + F_2 \cdot \left(x_r^G - x_{P\_\text{worst}}^G\right) \tag{6}$$

where $x_{P\_\text{best}}^G$ is selected from the best 10% individuals, $x_{P\_\text{worst}}^G$ is selected from the worst 10% individuals, and x_r^G is selected from the range between them. F_1 and F_2 are generated according to a uniform distribution in (0,1). EADE mutation strategy is combined with DE basic mutation strategy DE/rand/1/bin with a probability of 0.5.

Cr parameter is gradually adapted using a bool of Cr values (A). A has 11 Cr values [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 0.95]. The first 10% of the generations is used as a learning period. Each individual will select a Cr value from the bool (A) and advance with it until it fails to generate a better successor.

Adaptive DE with novel triangular mutation strategy

JADE and EADE could be considered as global directed mutation strategies. Both of them include the global best in the mutation to direct the search process. Adaptive DE with novel triangular mutation strategy (ANDE) was proposed to be a locally directed mutation strategy. It mimics the effect of gradient descent toward the nearest optima. The following equation describes ANDE mutation strategy:

$$v_i^G = \overline{x_c^G} + F_1 \cdot \left(x_{\text{best}}^G - x_{\text{better}}^G\right) + F_2 \cdot \left(x_{\text{best}}^G - x_{\text{worst}}^G\right) + F_3 \cdot \left(x_{\text{better}}^G - x_{\text{worst}}^G\right) \tag{7}$$

where x_{best}^G, x_{better}^G, and x_{worst}^G are three vectors that are randomly selected and then sorted from best to worst. F_1, F_2, and F_3 are generated according to the uniform distribution in (0,1). $\overline{x_c^G}$ is the convex combination vector defined as:

$$\overline{x_c^G} = w_1 \cdot x_{\text{best}}^G + w_2 \cdot x_{\text{better}}^G + w_3 \cdot x_{\text{worst}}^G \tag{8}$$

where w_1, w_2, and w_3 are real weights that satisfy $w_i > 0$ and $\sum_{i=1}^{3} w_i = 1$. ANDE uses the same Cr parameter adaptation used in EADE.

Modified multiple trajectory search

Three local search methods were introduced in multiple trajectory search (MTS) [4]. The first one of them, namely MTS-LS1, will be considered in this work. In each iteration, MTS-LS1 searches along one dimension. Candidate dimension is first subtracted by search range (SR) value. If this operation fails to generate a better successor, candidate dimension will be added by $0.5 \times$ SR. If this operation generates a better successor successfully, the new value will be retained. However, if MTS-LS1 does not improve the candidate vector, SR will be halved and the search starts over again. SR is initialized using $0.5 \cdot (U - L)$ where U is the upper bound of the problem and L is the lower bound. If SR reaches 1^{-15}, its value will be restored using $0.4 \cdot (U - L)$

In this work, a modified version of MTS-LS1 will be used. SR will be initialized using:

$$SR^d = rand.\left(\max\left(x^d\right) - \min\left(x^d\right)\right) \tag{9}$$

where $rand$ is a random number (0, 1). $\max\left(x^d\right)$and $\min\left(x^d\right)$ are the current minimum and maximum values of dimension d. In addition, we will change the upper limit for SR values to $0.2.(U - L)$ to endorse small movements along each dimension. If adding SR generates a better successor successfully, the search will continue with this direction by adding another SR value. This procedure will proceed until it fails to generate a better successor or reach the upper bound. The same is done for the subtraction procedure.

Hybridization framework

MLSHADE-SPA is a hybridization framework between population-based algorithms and local search algorithm. LSHADE-SPA, EADE, and ANDE are used as population-based algorithms for global exploration, while a modified version of MTS (MMTS) is used as a local search algorithm for local exploitation.

To deal with different types of problems in our framework efficiently, grouped dimensions, as well as all dimensions, are considered in the optimization process. Thus, the concept of divide and conquer was integrated into our framework. Theoretically, separable problems could be solved by solving each group of correlated dimensions separately, while non-separable problems need to consider all dimensions at once. The performance of the framework is affected by the suitable selection of correlated dimensions identifying mechanism. Additionally, prior information about the internal structure of the target problem is sometimes needed. Our approach deals with problems as a black box with no prior information about them. This is done by applying the divide and conquer process randomly.

Figure 1 shows MLSHADE-SPA pseudo code. The framework starts with a randomly generated population *P*. Then, the available computational resource (*max_nfes*) is divided into rounds. In our framework, we have 50 rounds (*round_nfes*), *round_nfes = max_nfes/50* as illustrated in line 7 from the pseudo code. In each round, population-based algorithms *EAs* will work during the first half of the computational resource (*round_nfes),* while local search algorithm *MMTS* will work during the second half (line 10 and 11).

During the first half of each round, LSHADE-SPA will start the optimization considering all dimensions (line 21) until it consumes its computational resources. After that, the concept of divide and conquer is applied. Accordingly, all dimensions will be randomly divided into three mutually exclusive groups (line 24), and each group will be assigned to LSHADE-SPA, EADE, and ANDE, respectively (lines 25–33) as subcomponent optimizers. This means that each one of population-based algorithms will try to optimize the problem by concentrating on the dimensions assigned to it. LSHADE-SPA will start this procedure working on one-third of the dimensions. LSHADE-SPA is followed by EADE algorithm, which will try to optimize the problem using another one-third of the dimensions. Finally, ANDE will concentrate on the remaining dimensions.

In the divide and conquer step, population-based algorithms will participate to gain more computational resources according to their performances. At the end of each round, the performance of each population-based algorithm will be calculated using the summation of differences between old and new fitness values. Then, the computational resource allocated to the algorithm will divide the computed summation. Thus, in each round, the performance of each population-based algorithm ω_{alg}^{r} will be calculated using:

$$\omega_{alg}^{r} = \frac{\sum_{i=1}^{n} f(x) - f(u)}{CC\_nfes_{alg}^{r}} \tag{10}$$

where f is the fitness function, x is the old individual, u is the offspring individual using algorithm alg, and NP is the total number of individuals that successfully generate new ones.

After that, the calculated performance of each population-based algorithm ω_{alg}^{r} will be used to calculate the improvement ratio imp_{alg}^{r} of each algorithm using:

$$imp_{alg}^{r} = \max\left(0.1, \frac{\omega_{alg}^{r}}{\sum_{alg=1}^{n\_alg} \omega_{alg}^{r}}\right) \tag{11}$$

```
Algorithm: MLSHADE-SPA algorithm

1  $N_r = N^{init}$; nfes=0; cc_n=50;
2  Initialize population $P_r = (x_{1,r}, ..., x_{N,r})$ randomly;
3  Initialize SPA parameters;
4  Initialize EADE parameters;
5  Initialize ANDE parameters;
6  Initialize MMTS parameters;
7  round_nfes=max_nfes/cc_n;
8  flag=0;
9  while nfes<max_nfes
10     EA_nfes=round(round_nfes/2);
11     MMTS_nfes=round(round_nfes/2);
12     SPA_nfes=round(EA_nfes/2);
13     if flag==0
14         SPA_CC_nfes=round(EA_nfes/6);
15         ANDE_CC_nfes=round(EA_nfes/6);
16         EADE_CC_nfes=round(EA_nfes/6);
17         flag=1;
18     else
19         Calculate CC_nfes according to Eq. (12);
20     end
21     [Pop,Fit] = SPA(SPA_nfes,Pop,Fit);
22     nfes=nfes+SPA_nfes;
23     Group_No=3;
24     CC_Ind=devide_Diminsions(Group_No, D);
25     Alg_Ind=find(CC_Ind==1)
26     [Pop,Fit,imp(1)]=SPA_CC(CC_nfes(1),Pop,Fit,Alg_Ind);
27     nfes=nfes+CC_nfes(1);
28     Alg_Ind=find(CC_Ind==2)
29     [Pop,Fit,imp(2)]=EADE_CC(CC_nfes(2),Pop,Fit,Alg_Ind);
30     nfes=nfes+CC_nfes(2);
31     Alg_Ind=find(CC_Ind==3)
32     [Pop,Fit,imp(3)]=ANDE_CC(CC_nfes(3),Pop,Fit,Alg_Ind);
33     nfes=nfes+CC_nfes(3);
34     [Pop,Fit] = MMTS(MMTS_nfes,Pop,Fit);
35     nfes=nfes+MMTS_nfes;
36     Calculate imp according to Eq. (11);
37     Calculate $N_{r+1}$ according to Eq. (13);
38     if $N_r < N_{r+1}$ then
39         Sort individuals in P based on their fitness
40         values and delete lowest $N_r - N_{r+1}$ members;
41     end
42 end
```

Fig. 1 MLSHADE-SPA algorithm

where ω_{alg}^{r} is the calculated performance of each population-based algorithms, n is the number of algorithms ($n = 3$), 0.1 is the minimum ratio assigned to each algorithm to maintain all algorithms to be executed simultaneously.

After calculating the improvement ratio imp_{alg}^{r} of each algorithm, computational resources allocated for each algorithm $CC\_nfes^{r}_{alg}$ will be updated (lines 13–20) according to the following equation:

$$CC\_nfes^{r}_{alg} = (1-\alpha) * CC\_nfes^{r-1}_{alg} + \alpha * 0.5 * EA\_nfes^{r} * imp^{r-1}_{alg} \quad (12)$$

where α is a learning rate (0.1 in our framework), $CC\_nfes^{r}_{alg}$ is the computational resource allocated for population-based algorithm *alg* at round *r*, EA_nfes^{r} is the computational resource allocated for all population-based algorithms at round *r*, and imp^{r-1}_{alg} is the improvement ratio during round $r-1$.

During the second half of each round, local search algorithm MMTS will be used to enhance the quality of best solution founded so far (line 34). MMTS will work on each dimension one by one. Thus, there is no need to split the work for MMTS.

Linear population size reduction (LPSR) was also integrated into MLSHADE-SPA. In LPSR, the population size will be decreased according to:

$$N_{r+1} = \text{round}\left[\left(\frac{N^{\text{init}} - N^{\min}}{0.5 * maxn\_fes}\right) * nfes + N^{\text{init}}\right] \quad (13)$$

where *nfes* is the current number of fitness evaluations, *max_nfes* is the maximum number of fitness evaluations, $N^{\text{init}} = 250$ is the initial population size, and $N^{\min} = 20$. According to the previous equation, *N* will reach the minimum number of individuals within the first half of *max_nfes*.

Experimental results

To evaluate the performance of MLSHADE-SPA, three performance analysis experiments were performed. First, MLSHADE-SPA was evaluated using CEC2010 and compared with 17 state-of-the-art algorithms. Second, CEC2013 is used to evaluate MLSHADE-SPA and compared with nine state-of-the-art algorithms. Finally, a parametric analysis was performed to study the effect of each component in MLSHADE-SPA. CEC2010 consists of 20 scalable optimization functions, while CEC2013 includes 15 functions. Each function in both benchmarks has different criteria. In general, they can be classified into four classes as shown in Table 1.

The dimensions (*D*) of all functions are 1000 except for two overlapping functions, *F*13 and *F*14 in CEC2013, where *D* is 905. The experiment was repeated 25 runs for each function and solution error measure $((x)-(x^*))$ was recorded at the end of each run, where is the best solution obtained and x^* is the well-known global optimum of each function. All problems have the global optimum within the given bounds,

Table 1 Classes of CEC2010 and CEC2013 benchmarks

Function class	CEC2010	CEC2013
Fully separable	*F*1–*F*3	*F*1–*F*3
Partially separable	*F*4–*F*18	*F*4–*F*11
Overlapping	–	*F*12–*F*14
Fully non-separable	*F*19–*F*20	*F*15

thus boundary correction was performed for each generated solution. The termination condition was set when the maximum number of evaluation *max_nfes* was reached. *max_nfes* was set to 3.0E+6 function evaluations (*fes*) according to CEC2010 and CEC2013 guidelines.

MLSHADE-SPA was implemented using MATLAB R2014a. The source code is available on the authors' website https://goo.gl/zw36nP. All experiments were performed using the Intel (R) Core (TM) i7-4790 CPU 3.6 GHz and 12 GB RAM.

Evaluation criteria

To evaluate the performance of MLSHADE-SPA, three evaluation criteria were used. The first is Formula One Score (FOS). Formula One Score was used in the latest LSGO competition (CEC2015). According to this criterion, the algorithms will be ranked from best to worst. Then, the top 10 ranks will get 25, 18, 15, 12, 10, 8, 6, 4, 2, and 1, respectively. Algorithms ranked more than top 10 will get zero. Maximum values of *R* indicate better performance.

The second and third are two non-parametric statistical hypothesis tests: Friedman test and multi-problem Wilcoxon signed-rank test using $\alpha = 0.05$ as a significance level [44].

As a null hypothesis, it is assumed that there is no significant difference between the mean results of the two samples, whereas the alternative hypothesis is that there is significance in the mean results of the two samples. Using the *p* value and comparing it with the significance level, the null hypothesis is rejected if the *p* value is less than or equal to the significance level of 5%. The *p* values under the significance level are shown in bold.

Wilcoxon's test uses R^+ as the sum of ranks for the functions in which the first algorithm outperforms the second algorithm and R^- as the sum of ranks for the opposite. Larger ranks indicate a larger performance discrepancy. In addition, one of three signs (+, −, and ≈) is assigned for the comparison of any two algorithms, where (+) sign means the first algorithm is significantly better than the second, (−) sign means the first algorithm is significantly worse than the second, and (≈) sign means that there is no significant difference between the two algorithms.

All the *p* values in this paper were computed using SPSS version 20.00.

Table 2 MLSHADE-SPA statistical results using CEC2010

Func.	Best	Worst	Median	Mean	Std.
$f1$	5.59E−24	9.54E−23	3.19E−23	4.01E−23	2.88E−23
$f2$	3.18E+01	6.27E+01	4.88E+01	4.70E+01	7.83E+00
$f3$	7.46E−14	1.14E−13	9.95E−14	9.84E−14	9.70E−15
$f4$	6.05E+10	2.92E+11	1.81E+11	1.84E+11	6.32E+10
$f5$	4.47E+07	7.96E+07	5.97E+07	6.07E+07	9.52E+06
$f6$	3.55E−09	7.34E−09	7.11E−09	6.60E−09	1.29E−09
$f7$	4.09E−07	3.01E−05	5.07E−06	8.08E−06	7.44E−06
$f8$	7.18E−11	1.20E+05	1.74E−06	4.43E+03	2.30E+04
$f9$	1.20E+07	2.08E+07	1.68E+07	1.65E+07	2.34E+06
$f10$	4.40E+03	5.56E+03	5.03E+03	5.01E+03	2.72E+02
$f11$	7.17E+01	1.18E+02	8.64E+01	8.96E+01	1.38E+01
$f12$	2.95E+01	6.47E+01	4.22E+01	4.35E+01	9.61E+00
$f13$	1.04E+01	3.13E+02	7.48E+01	9.95E+01	8.47E+01
$f14$	5.10E+07	7.20E+07	5.98E+07	6.02E+07	6.03E+06
$f15$	3.54E+03	3.94E+03	3.76E+03	3.77E+03	9.59E+01
$f16$	2.02E+02	2.79E+02	2.50E+02	2.49E+02	1.65E+01
$f17$	4.49E+02	8.46E+02	6.32E+02	6.16E+02	9.20E+01
$f18$	3.22E+02	1.08E+03	5.28E+02	5.73E+02	1.76E+02
$f19$	3.98E+05	5.03E+05	4.51E+05	4.51E+05	2.34E+04
$f20$	7.72E−04	3.39E+02	1.06E+02	1.26E+02	9.55E+01

Table 3 MLSHADE-SPA statistical results using CEC2013

Func.	Best	Worst	Median	Mean	Std.
$f1$	7.45E−24	2.41E−21	4.87E−23	1.94E−22	4.79E−22
$f2$	6.27E+01	9.55E+01	7.96E+01	7.89E+01	9.69E+00
$f3$	8.53E−14	1.14E−13	9.95E−14	9.96E−14	7.91E−15
$f4$	2.67E+08	2.02E+09	5.76E+08	6.90E+08	4.41E+08
$f5$	1.37E+06	2.20E+06	1.81E+06	1.80E+06	2.34E+05
$f6$	2.30E+00	5.45E+03	1.94E+01	1.40E+03	2.39E+03
$f7$	2.39E+04	1.14E+05	5.12E+04	5.31E+04	1.96E+04
$f8$	2.53E+12	2.49E+13	8.35E+12	9.77E+12	5.53E+12
$f9$	1.29E+08	2.02E+08	1.64E+08	1.61E+08	1.94E+07
$f10$	4.00E+02	9.46E+02	4.96E+02	6.56E+02	2.40E+02
$f11$	1.29E+07	8.69E+07	3.98E+07	4.04E+07	1.98E+07
$f12$	3.66E−09	2.86E+02	9.70E+01	1.04E+02	7.64E+01
$f13$	1.74E+07	2.05E+08	5.12E+07	7.21E+07	4.99E+07
$f14$	1.12E+07	2.41E+07	1.47E+07	1.52E+07	3.08E+06
$f15$	1.22E+07	4.44E+07	2.87E+07	2.76E+07	9.01E+06

Performance analysis using CEC2010 and CEC2013

Statistical results of MLSHADE-SPA using CEC2010 and CEC2013 benchmark are illustrated in Tables 2 and 3 respectively. The statistical results for all functions include best, worst, median, mean, and standard deviation calculated over 25 runs. Figure 2 illustrates the convergence behavior of MLSHADE-SPA using sample functions from each class in CEC2010: $f3$ as fully separable, $f8$ and $f11$ as partially separable, and $f20$ as fully non-separable.

The performance of MLSHADE-SPA was compared with reported results obtained from 26 algorithms. Using CEC2010, MLSHADE-SPA is compared with 16 algorithms illustrated in Table 4. While using CEC2013, it was compared with nine algorithms illustrated in Table 5. All of these algorithms were evaluated using the same benchmarks, and followed the same CEC2010 and CEC2013 guidelines.

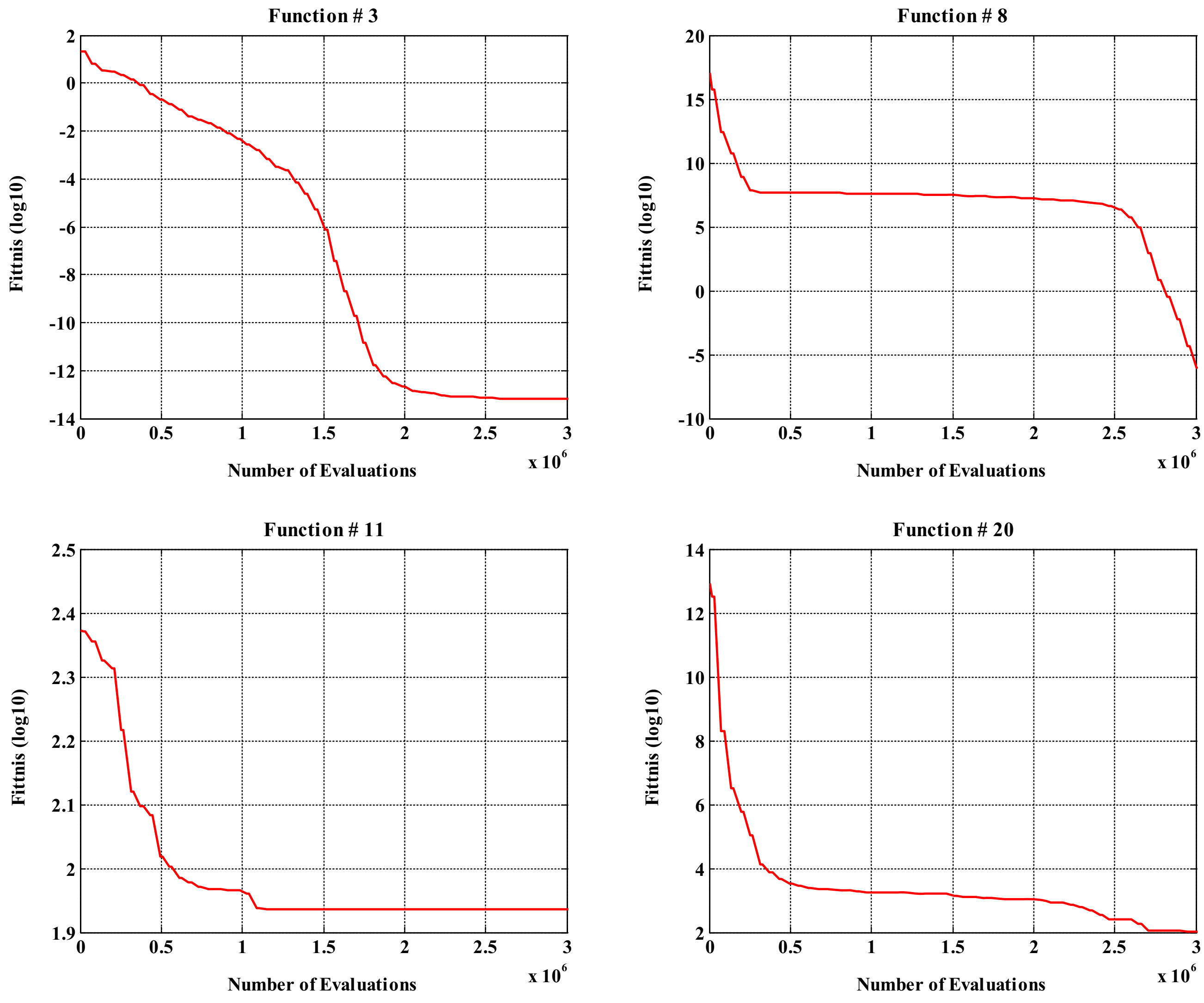

Fig. 2 Convergence behavior of MLSHADE-SPA using CEC2010 functions. $f3$ as fully separable, $f8$ and $f11$ as partially separable, and $f20$ as fully non-separable

Experimental comparisons between MLSHADE-SPA and these algorithms are illustrated in Tables 6 and 7 where the best results are distinguished in bold.

Tables 8 and 9 summarize the ranking of MLASHADE-SPA and the compared algorithms using Formula One Score (FOS). Tables 10 and 11 summarize the ranking obtained using Friedman's test. Finally, Tables 12 and 13 summarize the statistical analysis results of applying Wilcoxon's test between LSHADE_SPA and the compared algorithms.

Formula One Score (FOS)

As shown in Tables 8 and 9, MLSHADE-SPA has the second ranking among the compared algorithms using Formula One Score (FOS) for CEC2010 and the best ranking for CEC2013. Regarding CEC2010, the best ranking is MMO-CC with 236 points where MLSHADE-SPA gets 226 points. Comparing with the winners of previous CEC competitions: MA-SW-chains, the winner of CEC2010, gets 125 points, while MOS2012, the winner of CEC2012, gets 193 points. Using CEC2013, MLSHADE-SPA gets 254 points, followed by MOS2013, the winner of CEC2013 and CEC2015, and VGDE, with 218.5 and 194.5 points respectively.

Friedman test

According to Friedman test illustrated in Tables 10 and 11, MLSHADE-SPA has the best ranking for both CEC2010 and CEC2013 benchmarks. Using CEC2010, MLSHADE-SPA gets 5.3 points, MMO-CC gests 7.65, MA-SW-chains gets 7.70 points, and MOS2012 gets 8.45 points. While using

Table 4 List of compared algorithms using CEC2010

#	Algorithm	Published year
1	SDENS [23]	2010
2	jDElsgo [18]	2010
3	DECC-DML [21]	2010
4	MA-SW-chains [7]	2010
5	DASA [16]	2010
6	EOEA [11]	2010
7	LMDEa [22]	2012
8	jDEsps [19]	2012
9	MOS2012 [17]	2012
10	CCGS [20]	2012
11	DM-HDMR [28]	2014
12	HACC-D [29]	2014
13	DISCC [35]	2016
14	EADE [41]	2017
15	ANDE [42]	2017
16	SEE [38]	2018
17	MMO-CC [40]	2018

Table 5 List of compared algorithms using CEC2013

#	Algorithm	Published year
1	MOS2013 [25]	2013
2	SACC [26]	2013
3	DECC-CG [27]	2013
4	VGDE [32]	2014
5	IHDELS [33]	2015
6	CBCC3-DG2 [36]	2016
7	CRO [37]	2016
8	CCFR-I [39]	2017
9	CCFRI-DG2 [39]	2017

CEC2013, MLSHADE-SPA gets 3.13 points, and MOS2013 gets 3.43 points.

From the previous comparison, we can conclude that high ranking using Formula One Score (FOS) does not guarantee the same ranking using Friedman test. In Friedman test, the algorithms are ranked according to their mean rank with the same scale between any two successive positions. On the other hand, the scale is different for successive positions in Formula One Score (FOS). For example, the difference between any successive positions in Friedman test is just 1 point, while the difference between the successive positions in FOS is 7, 3, 3, 2, 2, 2, 2, 2, 1, 1, and 0 points, respectively. This means that using FOS, more weight will be given for the top positions.

Wilcoxon signed-rank test

Using Wilcoxon signed-rank test shown in Tables 12 and 13, we can see that MLASHADE-SPA obtains higher R^+ values than R^- in all cases for both benchmarks, except for IHDELS in CEC2013.

Using CEC2010 and according to Wilcoxon's test, the significant difference can be observed in 11 cases, which means that MLASHADE-SPA is significantly better than 11 algorithms out of 17 algorithms on 20 test functions. However, there is no significant difference in the remaining six cases. Furthermore, to be more precise, we can observe from Table 12 that MLSHADE-SPA is inferior to, equal to, superior to other algorithms in 86, 0, and 254 out of the total 340 cases. Thus, it can be concluded that the performance of MLSHADE-SPA is better than the performance of the compared algorithms in 74.71% of all cases, and it is outperformed by other compared algorithms in 25.29% of all cases.

Using CEC2013, the significant difference can be observed in three cases, which means that MLASHADE-SPA is significantly better than three algorithms out of nine algorithms. However, there is no significant difference in the remaining six cases. Furthermore, to be more precise, it is observable from Table 8 that MLSHADE-SPA is inferior to, equal to, superior to other algorithms in 32, 0, and 103 out of the total 135 cases. Thus, it can be concluded that the performance of MLSHADE-SPA is better than the performance of the compared algorithms in 76.30% of all cases, and it is outperformed by other compared algorithms in 23.70% of all cases.

Parametric analysis

In section III, we stated that MLSHADE-SPA framework consists of four components. These components are LSHADE-SPA, EADE, ANDE, and MMTS. To analyze the performance of MLSHADE-SPA, each of these components was evaluated separately using CEC2013. The experiment was repeated 25 runs for each component and the mean error values for all runs were recorded.

Table 14 illustrates the mean values of each component, and best values are marked in bold. MLSHADE-SPA was better than each of its individual components in nine functions and equal with LSHADE-SPA in one function, namely *f9*.

Using Formula One Score, we can see from Table 15 that MLSHADE-SPA outperforms each of its individual components with 327 points, followed by EADE, LSHADE-SPA, MMTS, and ANDE respectively.

Table 16 lists the average ranks according to Friedman test using CEC2013 benchmark problems. The p value com-

Table 6 Experimental comparisons between MLSHADE-SPA and state-of-the-art algorithms using CEC2010, FES = 3.0E+06

Func.	MLSHADE-SPA	EADE	ANDE	LMDEa	SDENS	jDElsgo	DECC-DML	MA-SW-chains	DISCC
f1	4.01E−23	4.70E−22	3.72E−26	1.35E−23	5.73E−06	8.86E−20	1.93E−25	2.10E−14	9.77E−25
f2	4.70E+01	4.16E+02	3.72E+02	6.97E+02	2.21E+03	1.25E−01	2.17E+02	8.10E+02	5.07E+02
f3	9.84E−14	6.25E−14	7.46E−14	6.44E−01	2.70E−05	3.81E−12	1.18E−13	7.28E−13	7.77E−14
f4	1.84E+11	1.08E+11	5.13E+11	2.08E+11	5.11E+12	8.06E+10	3.58E+12	3.53E+11	1.26E+12
f5	6.07E+07	8.79E+07	9.19E+07	6.62E+07	1.18E+08	9.72E+07	2.99E+08	1.68E+08	2.35E+08
f6	6.60E−09	1.90E+01	1.64E+00	2.63E−01	2.02E−04	1.70E−08	7.93E+05	8.14E+04	2.13E+06
f7	8.08E−06	2.11E−01	1.80E+00	2.45E−01	1.20E+08	1.31E−02	1.39E+08	1.03E+02	6.45E+05
f8	4.43E+03	**2.26E−04**	1.25E+07	3.61E−04	5.12E+07	3.15E+06	3.46E+07	1.41E+07	7.54E+07
f9	1.65E+07	3.67E+07	4.12E+07	2.64E+07	5.63E+08	3.11E+07	5.92E+07	1.41E+07	6.08E+07
f10	5.01E+03	2.62E+03	3.06E+03	2.80E+03	6.87E+03	2.64E+03	1.25E+04	2.07E+03	2.27E+03
f11	8.96E+01	1.14E+02	8.95E+01	1.19E+01	2.21E+02	2.20E+01	1.80E−13	3.80E+01	8.20E−01
f12	4.35E+01	2.80E+04	4.89E+04	1.83E+04	4.13E+05	1.21E+04	3.80E+06	3.62E−06	3.34E+04
f13	**9.95E+01**	1.01E+03	1.11E+03	5.95E+02	2.19E+03	7.11E+02	1.14E+03	1.25E+03	1.31E+03
f14	6.02E+07	1.46E+08	1.87E+08	8.63E+07	1.88E+09	1.69E+08	1.89E+08	3.11E+07	2.08E+08
f15	3.77E+03	3.18E+03	3.19E+03	5.63E+03	7.32E+03	5.84E+03	1.54E+04	2.74E+03	5.54E+03
f16	2.49E+02	3.00E+02	3.04E+02	3.87E+02	4.08E+02	1.44E+02	5.08E−02	9.98E+01	1.98E+01
f17	6.16E+02	1.52E+05	2.02E+05	2.14E+05	1.08E+06	1.02E+05	6.54E+06	1.24E+00	1.81E+05
f18	**5.73E+02**	2.26E+03	2.35E+03	1.68E+03	3.08E+04	1.85E+03	2.47E+03	1.30E+03	5.16E+03
f19	4.51E+05	1.29E+06	1.63E+06	4.42E+05	8.80E+05	2.74E+05	1.59E+07	2.85E+05	1.71E+06
f20	**1.26E+02**	2.10E+03	2.20E+03	1.38E+03	9.90E+02	1.53E+03	9.91E+02	1.07E+03	1.94E+03
Func.	DASA	EOEA	jDEsps	MOS2012	DM-HDMR	HACC-D	CCGS	SEE	MMO-CC
f1	1.52E−21	2.20E−23	4.10E−23	**0.00E+00**	2.34E+01	1.99E−27	1.83E−22	6.99E−11	**0.00E+00**
f2	8.48E+00	3.62E−01	1.10E+02	1.97E+02	4.36E+03	**1.43E−14**	4.44E−02	8.77E+03	1.43E+03
f3	7.20E−11	1.67E−13	1.30E−13	1.12E+00	1.67E+01	3.45E−14	1.91E−01	1.99E+01	**0.00E+00**
f4	5.05E+11	2.86E+12	8.15E+11	1.91E+10	6.96E+11	1.55E+12	1.79E+12	2.58E+11	**7.64E+06**
f5	6.20E+08	2.24E+07	7.71E+07	6.81E+08	1.45E+08	1.96E+08	**1.97E+07**	5.85E+08	3.34E+08
f6	1.97E+07	3.85E+06	5.58E−03	1.99E+07	1.63E+01	**3.55E−09**	2.88E+06	1.99E+07	5.77E−01
f7	7.78E+00	1.24E+02	5.77E+05	**0.00E+00**	2.91E+05	3.87E−07	1.37E+02	3.14E−02	2.41E+10
f8	4.98E+07	1.01E+07	1.52E+06	1.12E+06	4.41E+07	7.44E+07	2.81E+07	1.82E+06	2.63E+08
f9	3.60E+07	4.63E+07	2.31E+04	5.75E+06	5.20E+07	3.32E+07	5.53E+07	2.67E+07	**8.99E+01**
f10	**7.29E+03**	1.00E+03	1.85E+03	7.86E+03	4.49E+03	1.30E+04	4.74E+03	1.27E+04	1.63E+03
f11	1.98E+02	3.18E+01	1.94E−05	1.99E+02	1.10E+01	**7.82E−14**	2.99E+01	2.19E+02	2.99E+00
f12	1.78E+03	2.61E+04	1.57E+04	**0.00E+00**	1.97E+03	1.31E+06	5.35E+03	2.60E+02	**0.00E+00**
f13	1.21E+03	1.24E+03	1.86E+02	1.36E+03	3.35E+06	1.96E+03	1.51E+03	7.12E+02	3.05E+04
f14	1.00E+08	1.65E+08	3.85E+05	1.52E+07	3.41E+08	9.21E+07	1.35E+08	9.88E+07	**0.00E+00**
f15	1.45E+04	2.14E+03	5.50E+03	1.54E+04	5.95E+03	1.56E+04	**1.74E+03**	1.50E+04	2.05E+03
f16	3.97E+02	8.26E+01	4.97E+00	3.97E+02	1.24E−06	**1.95E−11**	3.11E+01	3.97E+02	8.87E+00
f17	1.03E+04	7.93E+04	5.52E+04	4.66E−05	4.03E+04	1.42E+06	1.48E+04	7.40E+03	**0.00E+00**
f18	4.92E+03	2.94E+03	9.73E+02	3.91E+03	8.40E+03	4.02E+03	3.13E+03	3.14E+03	3.37E+04
f19	8.34E+05	1.84E+06	8.00E+05	**3.41E+04**	1.71E+06	1.87E+07	5.93E+05	7.13E+05	1.54E+07
f20	1.13E+03	1.97E+03	8.79E+02	5.31E+02	2.45E+06	1.51E+03	1.31E+03	1.43E+03	1.10E+03

Table 7 Experimental comparisons between MLSHADE-SPA and state-of-the-art algorithms using CEC2013, FES = 3.0E+06

Func.	MLSHADE-SPA	MOS2013	DECC-CG	CBCC3-DG2	CCFR-IDG2	CCFR-I	CRO	IHDELS	VGDE	SACC
f1	1.94E−22	**0.00E+00**	2.03E−13	8.65E+05	2.00E−05	1.30E−05	1.84E+06	4.34E−28	**0.00E+00**	2.73E−24
f2	7.89E+01	8.32E+02	1.03E+03	1.41E+04	3.60E+02	**5.50E−01**	9.84E+02	1.32E+03	4.56E+01	7.06E+02
f3	**9.96E−14**	9.17E−13	2.87E−10	2.06E+01	2.10E+01	2.00E+01	2.01E+01	2.01E+01	3.98E−13	1.11E+00
f4	6.90E+08	1.74E+08	2.60E+10	**3.39E+07**	9.60E+07	4.50E+07	1.55E+10	3.04E+08	5.96E+08	4.56E+10
f5	**1.80E+06**	6.94E+06	7.28E+14	2.14E+06	2.80E+06	2.50E+06	2.38E+07	9.59E+06	3.00E+06	7.74E+06
f6	**1.40E+03**	1.48E+05	4.85E+04	1.05E+06	1.10E+06	1.10E+06	1.06E+06	1.03E+06	1.31E+05	2.47E+05
f7	5.31E+04	1.62E+04	6.07E+08	2.95E+07	2.00E+07	8.60E+06	2.78E+08	3.46E+04	**1.85E+03**	8.98E+07
f8	9.77E+12	8.00E+12	4.26E+14	6.74E+10	7.00E+10	**9.60E+09**	4.56E+14	1.36E+12	7.00E+14	1.20E+15
f9	**1.61E+08**	3.83E+08	4.27E+08	1.70E+08	1.90E+08	1.90E+08	5.27E+08	6.74E+08	2.31E+08	5.98E+08
f10	6.56E+02	9.02E+05	1.10E+07	9.28E+07	9.50E+07	9.50E+07	9.44E+07	9.16E+07	**1.57E+02**	2.95E+07
f11	4.04E+07	5.22E+07	2.46E+11	7.70E+08	4.00E+08	3.30E+08	2.91E+10	**1.07E+07**	7.52E+07	2.78E+09
f12	**1.04E+02**	2.47E+02	1.04E+03	5.81E+07	1.60E+09	6.00E+08	3.69E+03	3.77E+02	2.52E+03	8.73E+02
f13	7.21E+07	**3.40E+06**	3.42E+10	6.03E+08	1.20E+09	9.30E+08	5.33E+09	3.80E+06	1.36E+09	1.78E+09
f14	**1.52E+07**	2.56E+07	6.08E+11	1.11E+09	3.40E+09	2.10E+09	6.08E+10	1.58E+07	2.29E+10	1.75E+10
f15	2.76E+07	2.35E+06	6.05E+07	7.11E+06	9.80E+06	8.20E+06	1.88E+07	2.81E+06	3.44E+06	**2.01E+06**

Table 8 Ranks of 18 algorithms using CEC2010 according to FOS

Algorithm	FOS
MMO-CC	236
MLSHADE-SPA	226
MOS 2012	193
jDEsps	181
HACC-D	159
LMDEa	125
MA-SW-chains	125
jDElsgo	124
EADE	103
CCGS	98
EOEA	89
SEE	70
DECC-DML	69
ANDE	64
DISCC	50
DASA	41
DM-HDMR	40
SDENS	27

Table 9 Ranks of ten algorithms using CEC2013 according to FOS

Algorithm	FOS
MLSHADE-SPA	254
MOS2013	218.5
VGDE	194.5
IHDELS	171
CCFR-I	163.5
CBCC3-DG2	146
SACC	117
CCFR-IDG2	114.5
DECC-CG	84
CRO	52

puted through Friedman test was 0.00E+00. Thus, it can be concluded that there is a significant difference between the performances of the algorithms. Table 16 clearly shows that MLSHADE-SPA gets the first ranking, followed by EADE, LSHADE-SPA, MMTS, and ANDE, respectively.

Furthermore, Table 17 summarizes the statistical analysis results of applying Wilcoxon's test between MLSHADE-SPA and its individual components. We can see that MLSHADE-SPA obtains higher R^+ values than R^- values in comparison with its individual components. According to Wilcoxon's test, the significant difference can be observed in two cases only, namely MMTS and ANDE, which means that MLSHADE-SPA is better than MMTS and ANDE algorithms significantly. Moreover, MLSHADE-SPA is inferior to, equal to, superior to its individual components in 8, 1, and 51 out of the total 60 cases. Thus, it can be concluded that the performance of MLSHADE-SPA is better than the performance of its individual components in 85% of all cases, and it is just outperformed in 15% of all cases.

Based on the previous experimental and parametric analysis, we can conclude that the performance of MLSHADE-SPA framework significantly outperforms its four individual components.

Table 10 Ranking of MLSHADE-SPA and other algorithms according to Friedman test using CEC2010

Algorithm	Ranking
MLSHADE-SPA	5.30
jDEsps	6.10
MMO-CC	7.65
jDElsgo	7.70
MA-SW-chains	7.70
LMDEa	7.90
MOS2012	8.45
EADE	8.80
CCGS	9.35
EOEA	9.55
HACC-D	9.85
ANDE	9.90
DASA	11.25
SEE	11.38
DISCC	11.63
DECC-DML	11.78
DM-HDMR	12.48
SDENS	14.25

Table 11 Ranking of MLSHADE-SPA and other algorithms according to Friedman test using CEC2013

Algorithm	Ranking
MLSHADE-SPA	3.13
MOS2013	3.43
VGDE	4.30
IHDELS	4.83
CCFR-I	5.17
CBCC3-DG2	5.60
CCFRI-DG2	6.37
SACC	6.40
DECC-CG	7.47
CRO	8.30

Table 12 Results of Wilcoxon's test between MLSHADE-SPA and other algorithms at 0.05 significance level using CEC2010

Algorithms	R^+	R^-	p value	+	≈	−	Dec.
jDEsps	149	61	0.100	15	0	5	≈
MMO-CC	113	97	0.765	9	0	11	
jDElsgo	145	65	0.135	14	0	6	≈
ma-sw-chains	117	93	0.654	11	0	9	≈
LMDEa	165	45	**0.025**	15	0	5	+
MOS2012	126	84	0.433	12	0	8	≈
EADE	155	55	0.062	15	0	5	≈
CCGS	162	48	**0.033**	14	0	6	+
EOEA	159	51	**0.044**	13	0	7	+
ANDE	185	25	**0.003**	15	0	5	+
HACC-D	182	28	**0.004**	13	0	7	+
DASA	206	4	**0.000**	19	0	1	+
SEE	210	0	**0.000**	20	0	0	+
DISCC	191	19	**0.001**	15	0	5	+
DECC-DML	201	9	**0.000**	17	0	3	+
DM-HDMR	195	15	**0.001**	17	0	3	+
SDENS	210	0	**0.000**	20	0	0	+

Table 13 Results of Wilcoxon's test between MLSHADE-SPA and other algorithms at 0.05 significance level using CEC2013

Algorithms	R^+	R^-	p value	+	≈	−	Dec.
MOS2013	62	58	0.910	9	0	6	≈
VGDE	85	35	0.156	9	0	6	≈
IHDELS	56	64	0.820	8	0	7	≈
CCFR-I	83	37	0.191	11	0	4	≈
CBCC3-DG2	86	34	0.140	12	0	3	≈
CCFRI-DG2	88	32	0.112	12	0	3	≈
SACC	112	8	**0.003**	13	0	2	+
DECC-CG	120	0	**0.001**	15	0	0	+
CRO	114	6	**0.002**	14	0	1	+

Finally, MLSHADE-SPA is considered as a modified version of LSHADE-SPA. Since LSHADE-SPA was evaluated using CEC2017, MLSHADE-SPA was also evaluated using the same benchmark. Table 18 illustrates mean values obtained using MLSHADE-SPA for 10, 30, 50 and 100 dimensions.

Comparing with LSHADE-SPA, Table 19 illustrates the ranks according to Friedman test. We can clearly observe that LSHADE-SPA is significantly better than MLSHADE-SPA. We also applied the performance assessment based on score metric, which is recently defined for the CEC2017 competition [45]. The evaluation method for both algorithms is based on a score of 100%, where 50% is for *SE* as the summation of error values for all dimensions, and 50% is for *SR* as rank-based evaluation for each problem in each dimension. Table 20 illustrates *score1*, *score2*, and score achieved by both algorithms. We can clearly see that LSHADE-SPA was also better than MLSHADE-SPA according to CEC2017 metric with 100% score, while MLSHADE-SPA gets just 53.56%.

According to the previous comparisons, we see that the performance of LSHADE-SPA is better than MLSHADE-SPA in small scale and worse in large scale. There is no contradiction here since it known that an algorithm exhibiting a good performance on low-dimensional problems may degrade as the problem size increases. This explains the performance behavior of LSHADE-SPA. On the

Table 14 Mean values of MLSHADE-SPA, LSHADE-SPA, EADE, ANDE and MMTS using CEC2013

Func.	LSHADE-SPA	EADE	ANDE	MMTS	MLSHADE-SPA
$f1$	6.26E−18	6.62E−14	7.34E+06	4.64E−03	**1.94E−22**
$f2$	9.11E+02	2.92E+02	9.85E+03	**3.69E−07**	7.89E+01
$f3$	1.16E+00	8.09E−13	9.36E+00	1.18E−05	**9.96E−14**
$f4$	3.51E+09	2.07E+09	3.30E+11	1.11E+11	**6.90E+08**
$f5$	1.63E+06	**1.25E+06**	7.48E+06	1.83E+07	1.80E+06
$f6$	5.43E+03	5.47E+03	**2.28E+01**	9.83E+05	1.40E+03
$f7$	5.78E+06	3.59E+06	4.20E+09	1.57E+09	**5.31E+04**
$f8$	**2.08E+12**	2.04E+13	6.81E+15	8.40E+15	9.77E+12
$f9$	**1.61E+08**	8.60E+07	5.28E+08	1.51E+09	**1.61E+08**
$f10$	2.91E+04	8.46E+02	7.60E+02	8.01E+07	**6.56E+02**
$f11$	2.55E+08	2.21E+08	3.80E+11	3.71E+11	**4.04E+07**
$f12$	3.35E+03	2.38E+03	1.16E+11	1.44E+03	**1.04E+02**
$f13$	4.02E+08	4.04E+08	1.92E+11	1.13E+10	**7.21E+07**
$f14$	2.15E+08	6.88E+08	1.65E+12	3.14E+11	**1.52E+07**
$f15$	3.37E+06	**2.76E+06**	1.72E+08	3.88E+08	2.76E+07

Table 15 Ranking of MLSHADE-SPA and other algorithms according to FOS using CEC2013

Algorithm	FOS
MLSHADE-SPA	327
EADE	264
LSHADE-SPA	238
MMTS	190
ANDE	181

Table 16 Ranking of MLSHADE-SPA and its individual components according to Friedman test using CEC2013

Algorithm	Ranking
MLSHADE-SPA	1.57
EADE	2.40
LSHADE-SPA	2.77
MMTS	4.00
ANDE	4.27

Table 17 Results of Wilcoxon's test between MLSHADE-SPA and its individual components at 0.05 significance level using CEC2013

Algorithms	R^+	R^-	p value	+	≈	−	Dec.
LSHADE-SPA	75	30	0.158	11	1	3	≈
EADE	94	26	0.053	12	0	3	≈
MMTS	117	3	**0.001**	14	0	1	+
ANDE	117	3	**0.001**	14	0	1	+

other hand, an algorithm exhibiting a good performance on high-dimensional problems does not guarantee a good performance on low-dimensional problems. This observation is

Table 18 Mean values of MLSHADE-SPA for the dimensions 10, 30, 50, and 100 using CEC2017

Func.	10D	30D	50D	100D
$f1$	2.49E−05	0.00E+00	1.58E−06	0.00E+00
$f3$	0.00E+00	4.03E−01	2.62E+03	6.24E+04
$f4$	2.34E−02	1.73E+01	2.72E+01	9.98E+01
$f5$	3.36E+00	3.16E+01	9.17E+01	1.85E+02
$f6$	0.00E+00	2.21E−08	8.25E−08	7.32E−07
$f7$	1.40E+01	7.24E+01	1.21E+02	2.67E+02
$f8$	3.39E+00	1.31E+02	7.72E+01	1.85E+02
$f9$	0.00E+00	2.50E+02	3.51E−03	3.48E−01
$f10$	1.08E+02	3.24E+03	4.75E+03	1.28E+04
$f11$	2.90E−01	1.06E+01	4.32E+01	3.03E+02
$f12$	4.62E+02	1.34E+04	1.57E+05	4.15E+05
$f13$	5.26E+00	7.30E+01	2.30E+02	1.55E+03
$f14$	6.05E−01	3.75E+01	1.67E+02	2.59E+04
$f15$	2.96E−01	1.82E+01	7.81E+01	1.90E+02
$f16$	4.57E−01	3.14E+02	7.06E+02	2.40E+03
$f17$	3.65E−01	4.17E+01	4.71E+02	1.64E+03
$f18$	8.06E−01	4.50E+03	1.26E+04	9.08E+04
$f19$	3.95E−02	1.55E+01	4.39E+01	2.41E+02
$f20$	2.45E−02	6.87E+01	3.66E+02	1.79E+03
$f21$	1.04E+02	2.33E+02	2.73E+02	2.43E+02
$f22$	8.78E+01	1.00E+02	2.83E+03	1.34E+04
$f23$	2.99E+02	3.76E+02	4.97E+02	6.74E+02
$f24$	1.91E+02	4.48E+02	5.57E+02	1.08E+03
$f26$	3.99E+02	3.87E+02	5.08E+02	7.45E+02
$f27$	2.90E+02	1.15E+03	1.80E+03	4.99E+03
$f28$	3.91E+02	5.00E+02	5.20E+02	6.19E+02
$f29$	2.91E+02	3.31E+02	4.66E+02	5.22E+02
$f30$	2.42E+02	4.53E+02	4.07E+02	1.93E+03

Table 19 Ranking of LSHADE-SPA and MLSHADE-SPA according to Friedman test using CEC2017

Algorithm	Ranking
LSHADE-SPA	1.21
MLSHADE-SPA	1.79

Table 20 Comparison between MLSHADE-SPA and LSHADE-SPA according to evaluation criteria used in CEC2017

Algorithm	Score 1 (%)	Score 2 (%)	Score (%)	*R*
LSHADE-SPA	50	50	100	1
MLSHADE-SPA	21.72	31.84	53.56	2

a practical example of the “no free lunch” theorem [46]. As an example, the performance of MOS2011 and MOS2013, the winners of recently LSGO CEC competitions, were also evaluated using CEC2017 and their ranks were 9th and 10th, respectively.

Conclusion

An improved framework for solving LSGO is introduced in this paper. According to the experimental results, MLSHADE-SPA significantly outperforms many state-of-the-art algorithms.

Looking at the results achieved using our algorithm and other algorithms shows that there is an opportunity to achieve better results for both benchmarks. MLSHADE-SPA opens promising improvement research points. Simplifying the target problem using the concept of divide and conquer without any prior knowledge about the internal structure of the problem is an interesting research field. On the other hand, intelligently hybridization of appropriate optimizers is still a challenging task and a promising research field. Finally, according to the performance of competitive algorithms, it seems that using a local search algorithm along with a population-based one is a suitable choice for solving LSGO problems efficiently.

References

1. Goh SK, Abbas HA, Tan KC (2015) Optimization of big data 2015 competition. http://www.husseinabbass.net/BigOpt.html
2. Chen S, Montgomery J, Bolufé-Röhler A (2015) Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. Appl Intell 42(3):514–526
3. Omidvar M (2015) Cooperative co-evolutionary algorithms for large-scale optimization. RMIT University, Melbourne
4. Tseng LY, Chen C (2008) Multiple trajectory search for large scale global optimization. Evolut Comput CEC2008:3052–3059
5. Zhao SZ, Suganthan PN, Das S (2011) Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. Soft Comput 15(11):2175–2185
6. Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE transactions on evolutionary computation
7. Molina D, Lozano M, Herrera F (2010) MA-SW-Chains: memetic algorithm based on local search chains for large scale continuous global optimization. In: Evolutionary computation CEC2010 IEEE Congress on, pp 1–8
8. Hansen N, Müller SD, Koumoutsakos P (2003) Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolut Comput 11:1–18
9. Solis FJ, Wets RJ (1981) Minimization by random search techniques. Math Oper Res 6:19–30
10. Tang K, Li X, Suganthan P, Yang Z, Weise T (2010) Benchmark functions for the CEC2010 special session and competition on large scale global Optimization
11. Wang Y, Li B (2010) Two-stage based ensemble optimization for large-scale global optimization. In: Evolutionary computation (CEC), 2010 IEEE Congress on IEEE, pp 1–8
12. Wang Y, Li B (2009) A self-adaptive mixed distribution based univariate estimation of distribution algorithm for large scale global optimization. In nature-inspired algorithms for optimisation. Springer, Berlin, pp 171–198
13. Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of IEEE congress on evolutionary computation, pp 1785–1791
14. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evolut Comput 6(2):182–197
15. Wang Y, Li B (2009) Multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization. Memetic Comput 2(1):1–22
16. Korosec P, Tashkova K, Silc J (2010) The differential ant-stigmergy algorithm for large-scale global optimization. In: IEEE congress on evolutionary computation (CEC 2010), pp 4288–95
17. LaTorre A, Muelas S, Peña JM (2012) Multiple offspring sampling in large scale global optimization. In: IEEE congress on evolutionary computation (CEC 2012), Brisbane, Australia, pp 964–71
18. Brest J, Zamuda A, Boškovi´c B, Fister I, Mauˇcec MS (2010) Large scale global optimization using self-adaptive differential evolution algorithm”, In: IEEE World Congress on computational intelligence, pp 3097–3104
19. Brest J, Bošković B, Zamuda A, Fister I, Maučec MS (2012) Self-adaptive differential evolution algorithm with a small and varying population size. In: IEEE congress on evolutionary computation (CEC 2012), Brisbane, pp 1–8
20. Zhang K, Li B (2012) Cooperative coevolution with global search for large scale global optimization. In: IEEE congress on evolutionary computation (CEC 2012), Brisbane, pp 1–7
21. Omidvar MN, Li XD, Yao X (2010) Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In: 2010 IEEE Congress on evolutionary computation (CEC), pp 1762–1769
22. Takahama T, Sakai S (2012) Large scale optimization by differential evolution with landscape modality detection and a diversity archive. In: Proceedings of 2012 IEEE Congress on evolutionary computation, pp 2842–2849

23. Wang H, Wu Z, Rahnamayan S, Jiang D (2010) Sequential DE enhanced by neighborhood search for large scale global optimization. In: Proceedings of the IEEE Congress on evolutionary computation, pp 4056–4062
24. Li X, Tang K, Omidvar MN, Yang Z, Quin K (2013) Benchmark functions for the CEC2013 special session and competition on large scale global optimization", RMIT University
25. LaTorre A, Muelas S, Peña JM (2013) Large scale global optimization: experimental results with MOS-based hybrid algorithms. In: 2013 I.E. congress on evolutionary computation (CEC 2013), Cancún, Mexico, pp 2742–9
26. Wei F, Wang Y, Huo Y (2013) Smoothing and auxiliary functions based cooperative coevolution for global optimization. In: IEEE congress on evolutionary computation (CEC 2013), pp 2736–41
27. Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 78(15):2985–2999
28. Mahdavi ME, Rahnamayan S (2014) Cooperative co-evolution with a new decomposition method for large-scale optimization. IEEE congress on evolutionary computation (CEC), Beijing, China
29. Ye S, Dai G, Peng L, Wang M (2014) A hybrid adaptive coevolutionary differential evolution algorithm for large-scale optimization. In: Evolutionary computation (CEC), 2014 IEEE Congress on IEEE, pp 1277–1284
30. Yang Z, Tang K, Yao X (2008) Self-adaptive differential evolution with neighborhood search. In: Proc. of IEEE World Congress on Computational Intelligence, pp 1110–1116
31. Molina D, LaTorre A, Herrera F (2018) An insight into bio-inspired and evolutionary algorithms for global optimization: review, analysis, and lessons learnt over a decade of competitions. Cognit Comput 10:1–28
32. Wei F, Wang Y, Zong T (2014) Variable grouping based differential evolution using an auxiliary function for large scale global optimization. In: IEEE congress on evolutionary computation (CEC 2014)
33. Molina D, Herrera F (2015) Iterative hybridization of DE with local search for the CEC2015 special session on large scale global optimization. IEEE Congress on evolutionary computation (CEC 2015)
34. Zhu C, Byrd RH, Lu P, Nocedal J (1997) Algorithm 778: L-BFGS-B: fortran subroutines for large-scale bound-constrained optimization. ACM Trans Math Softw (TOMS) 23(4):550–560
35. Dai G, Chen X, Chen L, Wang M, Peng L (2016) Cooperative coevolution with dependency identification grouping for large scale global optimization. In: Evolutionary computation (CEC), 2016 IEEE Congress on IEEE, pp 5201–5208
36. Omidvar MN, Kazimipour B, Li X, Yao X (2016) CBCC3-A contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance. In: Evolutionary computation (CEC), 2016 IEEE Congress on IEEE, pp 3541–3548
37. Salcedo-Sanz S, Camacho-Gómez C, Molina D, Herrera F (2016) A Coral reefs optimization algorithm with substrate layers and local search for large scale global optimization", In: IEEE congress on evolutionary computation (CEC 2016)
38. Yang P, Tang K, Yao X (2018) Turning high-dimensional optimization into computationally expensive optimization. IEEE Trans Evolut Comput 22(1)
39. Yang M, Omidvar MN, Li C, Li X, Cai Z, Kazimipour B, Yao X (2017) Efficient resource allocation in cooperative co-evolution for large-scale global optimization. IEEE Trans Evol Comput 21(4):493–505
40. Peng X, Jin Y, Wang H (2018) Multimodal optimization enhanced cooperative coevolution for large-scale optimization. IEEE Trans Cybern 99
41. Mohamed AW (2017) Solving large-scale global optimization problems using enhanced adaptive differential evolution algorithm. Complex Intell Syst 3(4):205–231
42. Mohamed AW, Almazyad AS (2017) Differential evolution with novel mutation and adaptive crossover strategies for solving large scale global optimization problems. Appl Comput Intell Soft Comput 7974218
43. Mohamed AW, Hadi AA, Fattouh AM, Jambi KM (2017) LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. In: Evolutionary computation (CEC), 2017 IEEE Congress on IEEE, pp 145–152
44. García S, Molina D, Lozano M, Herrera F (2009) 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 15:617–644
45. Awad NH, Ali MZ, Liang JJ, Qu BY, Suganthan PN (2016) Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. Technical report, Nanyang Technological University, Singapore
46. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82