Large Initial Population and Neighborhood Search incorporated in LSHADE to solve CEC2020 Benchmark Problems

Partha P. Biswas
Advanced Digital Sciences Center, Singapore
partha.b@adsc-create.edu.sg

Ponnuthurai N. Suganthan
School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore
epnsugan@ntu.edu.sg

***Abstract*—The single objective multimodal bound-constrained optimization problems in CEC (IEEE Congress on Evolutionary Computation) competitions pose tremendous challenges to the researchers in finding the global optimum. This paper introduces the orthogonal array-based initialization of population and neighborhood search strategy in LSHADE (linear population reduction technique in success history based adaptive differential evolution) algorithm to solve the CEC2020 competition problems. In the novel algorithm, named as O-LSHADE, a large number of population members have been initialized in the search space using the orthogonal design. Thereafter, a small Euclidean neighborhood is defined for each member in the population. The adaptive mutation and crossover are performed within the neighborhood during first phase, i.e., predominantly exploration phase of the algorithm. The process ensures that the search space is adequately explored by the algorithm as the neighborhoods are not mixed and each neighborhood performs the mutation and crossover operations independently. At a later stage of the algorithm, a step reduction in population is done to perform the operations like an adaptive differential evolution. The proposed algorithm is found efficient, effective and robust when applied to the CEC2020 benchmark problems for single objective bound constrained optimization. In most cases, the algorithm could achieve the best error value of less than 100. This implies that the method could guide the search process towards the global basin.**

***Index Terms*—Differential evolution, LSHADE algorithm, orthogonal array, neighborhood search, numerical optimization.**

I. Introduction

Single objective real parameter optimization problems are prevalent in real world in the fields of engineering, science and research. In the last few decades, population-based evolutionary algorithms have been widely applied for solving real parameter optimization problems owing to their competitive performance, and simplicity in understanding and implementation. Differential evolution (DE) is one such evolutionary algorithm, first proposed by Storn and Price [1]. The algorithm works on three simple stages of operation – mutation, crossover and selection, on a randomly generated pool of population members in quest of the global optimum. Not long thereafter, DE has been able to establish itself as a powerful, robust and efficient technique in solving diverse nature of numerical optimization problems. However, original DE algorithm has fixed control parameters, a strategy which has been found wanting in dealing with the ever-increasing complexity of single objective bound constrained optimization problems. Over the years, researchers have proposed several variants of advanced DE algorithms, a genre of which focuses primarily on the adaptation strategies of the control parameters and/or mutation strategies (e.g., [2-5]) to enhance the exploration and exploitation capabilities of the algorithm. Another track of performance improvement of DE algorithm is by hybridizing it with other evolutionary algorithms or local search methods [6-7]. The improvements suggested in these algorithms are intended to efficiently solve the CEC benchmark problems which are extensively followed by the researchers across the globe to test their algorithms. Though the inception of CEC competitions or special sessions can be traced back to the year 2005, we would focus here on the competitions in this decade. A cursory glance at the competition results [8] reveals that except in the year 2013, an advanced variant of the DE algorithm has been one of the top performing algorithms in all CEC competitions in this decade. Indeed, variants of the LSHADE algorithm were the winners in all competitions in four consecutive years starting from the year 2014. The performance of the algorithm is superior not only for benchmark problems but also for real world optimization problems of discrete variables [9], continuous variables [10-11] and a combination of both [12].

The development of LSHADE algorithm had started based on another popular adaptive DE algorithm called JADE [3]. In JADE, a novel current/to/pbest mutation strategy with optional external archive and an adaptation technique of control parameters such as mutation (or scale) factor (F) and crossover rate (CR) based on statistical distributions were proposed. SHADE algorithm [4] tweaked the control parameter adaptation technique of JADE and updated those parameters relying on success history of previous generations in the search process. LSHADE [5], an extension of SHADE algorithm, suggested the linear reduction of population size in successive generations. In algorithm LSHADE-EpSin [13], a sinusoid function was introduced at early stage of exploration for adaptation of the mutation factor (F). In the same vein, LSHADE-*cn*EpSin [14] incorporated covariance matrix learning for the crossover operator (CR). iLSHADE [15] enforced some limitations on F and CR values during early exploration stage, while jSO [16] utilized weighted adaptive values of F in the mutation strategy. Hybridization of

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

modified control parameter adaptation technique with covariance matrix adaptation of evolutionary strategy was proposed in LSHADE-SPACMA [17]. Rank-based selection pressure strategy in LSHADE (LSHADE-RSP) [18] and distance-based parameter adaptation technique (DbL_SHADE) [19] are some other variants of the LSHADE algorithm. Readers may please refer to the article [20] for a summary of these algorithms. Very recently, Mohamed *et. al.* [21] proposed a new mutation strategy for SHADE and LSHADE variants.

The literature review on the improvement of LSHADE algorithm reveals that researchers have focused predominantly on the adaptation of a couple of control parameters, F and CR. The third control parameter population size (Np) has received little attention in the process, especially during the initialization of the population members. The 'random uniform' generation of members does not guarantee uniform distribution and enough coverage of the search space. Moreover, the diversity of the population members is easily lost in the greedy selection process. In this article, we utilize the orthogonal array-based design [22] for uniform creation of a large number of population members in the solution space. Subsequently, we perform the DE operations (i.e., mutation, crossover and selection) only within the neighborhood of each member for a greater part of the iteration process. In later half, we reduce the population size drastically and the members with the superior fitness values survive. All these remaining members participate in DE operations as greedy selection is employed.

In rest of the paper, the proposed algorithm O-LSHADE is explained step-by-step in section II. Section III presents the simulation results for the algorithms on CEC2020 benchmark problems. Finally, the paper ends with concluding remarks and potential future works in section IV.

II. Algorithm O-LSHADE

In this section, we describe our proposed O-LSHADE algorithm starting with the orthogonal array-based approach to generate the initial population members.

A. Construction of the orthogonal arrays

Prior to solving an optimization problem, we usually have limited information on the location of the global optimum. Therefore, it is desirable to have candidate solutions spread uniformly in the search space to facilitate the exploration process in pursuit of the optimal point. An orthogonal array specifies some combinations which are uniformly scattered over the space of all possible combinations. So, orthogonal design is an effective method to create a population with good coverage of the defined search space. Here, we design a special class of orthogonal arrays using a simple permutation method [22]. We denote Q as the number of levels for the array, N as the number of factors (loosely termed as columns here), and J as a positive integer such that the number of rows for the array is Q^J and it satisfies following condition:

$$N = \frac{Q^J - 1}{Q - 1} \tag{1}$$

We denote the orthogonal array as A, the entry at i-th row and j-th column as $A_{[i,j]}$, and the complete j-th column as $A_{[:,j]}$. The array has basic columns, defined as $A_{[:,j]}$ for $j = 1, 2, \frac{Q^2-1}{Q-1} + 1, \ldots, \frac{Q^{J-1}-1}{Q-1} + 1$. All the remaining columns are non-basic columns. We construct the basic columns first, followed by the non-basic columns as per the Algorithm 1:

Algorithm 1: Orthogonal array construction

Step 1: Construction of basic columns

for $k = 1$ to J **do**

$$j = \frac{Q^{k-1} - 1}{Q - 1} + 1;$$

for $i = 1$ to Q^J **do**

$$A_{[i,j]} = \left[\frac{i-1}{Q^{J-k}}\right] mod\ Q$$

where mod is the modulus after division, e.g., $5\ mod\ 3 = 2$, $7\ mod\ 3 = 1$, etc.

Step 2: Construction of non-basic columns

for $k = 2$ to J **do**

$$j = \frac{Q^{k-1} - 1}{Q - 1} + 1;$$

for $s = 1$ to $j - 1$ **do**

for $t = 1$ to $Q - 1$ **do**

$$A_{[:,j+(s-1)(Q-1)+t]} = \left(A_{[:,s]} * t + A_{[:,j]}\right) mod\ Q$$

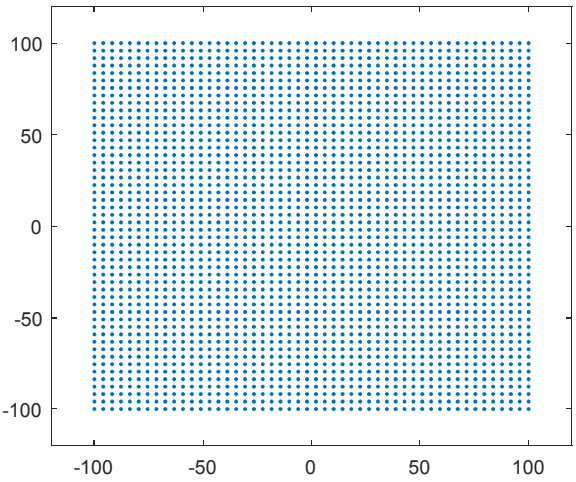

Fig. 1: Two-dimensional visualization of the initialized population members

B. Initial population and neighborhood

The optimization problems have defined fixed dimension (say D). However, the orthogonal arrays discussed above can construct N columns strictly adhering to Eq. (1). There may not

exist Q and J that satisfies $D = N$. To overcome the situation, we select a value of J such that it can construct the orthogonal array with $N \geq D$. Thereafter, we delete the last $(N - D)$ columns to obtain the orthogonal array with D factors. As our search range is within [-100,100], we scale the elements in each column to get the matrix of initial population members. In order to visualize the initialized members in the search space, we present a two-dimensional plot of 2500 members in Fig. 1 by considering only the first 2 elements of each population member. The two-dimensional visualization plot of the initialized members shows adequate coverage of the solution space.

We compute the Euclidean distance between each pair of the initialized population members to define the neighborhood. The neighborhood of each member is its nearest n_b members according to the distance calculation. The neighborhood remains unchanged throughout the iteration process of the algorithm to reduce any extra computational burden. To elaborate, a target vector (x_i) has fixed n_b indices of population members as its neighborhood. As DE strategic operations are restricted within the neighborhood in early stage, we presume that the new members are evolved mostly in the local search space. Further, a large pool of population members (say, Np_{init}) and small neighborhood size (n_b) ensure that the search operation is focused only in the vicinity of each member in this phase.

C. Evolution during first phase

We dedicate about 60% of the total number of allowable function evaluations for first phase with the primary intention of exploration. In this stage, we restrict the DE operations (i.e., mutation and crossover) within the neighborhood. The selected mutation strategy of 'DE/current to best/1' perturbs the target vector (x_i) with the best member in its neighborhood. At g-th generation (iteration), the mutation strategy is written as:

$$v_i^{(g)} = x_i^{(g)} + F_i^{(g)}.\left(x_{i\_nbst}^{(g)} - x_i^{(g)}\right) + F_i^{(g)}.\left(x_{R_{nb1}^i}^{(g)} - x_{R_{nb2}^i}^{(g)}\right) \quad (2)$$

where superscript 'g' signifies g-th generation, $v_i^{(g)}$ is the mutant vector, $x_i^{(g)}$ is the target vector, $x_{i\_nbst}^{(g)}$ is the best member among the n_b neighborhood of $x_i^{(g)}$, R_{nb1}^i & R_{nb2}^i are the mutually exclusive integer within the neighborhood $[1, n_b]$. $F_i^{(g)}$ is the globally adapted mutation (scale) factor associated with $x_i^{(g)}$, the adaptation process of which is described later in this section. The values of all elements of $v_i^{(g)}$ must be within the defined range $[x_{min}, x_{max}]$. If any element falls outside the search range boundary, the same is corrected as:

$$v_{i,j}^{(g)} = \begin{cases} x_{min,j} \text{ if } v_{i,j}^{(g)} < x_{min,j} \\ x_{max,j} \text{ if } v_{i,j}^{(g)} > x_{max,j} \end{cases} \quad (3)$$

Please note that in our numerical optimization problems, all $x_{min,j} = -100$ and $x_{max,j} = 100$ for $j \in [1, D]$, D denotes the dimension of the problem. Following the mutation operation, we perform the widely used binomial crossover of DE. The trial or offspring vector $\left\{u_i^{(g)} = \left(u_{i,1}^{(g)}, u_{i,2}^{(g)}, \ldots\ldots, u_{i,D}^{(g)}\right)\right\}$ is obtained by crossover operation with the mutant vector $v_i^{(g)}$. The elementwise operation can be expressed as:

$$u_{i,j}^{(g)} = \begin{cases} v_{i,j}^{(g)} \text{ if } j = j_{rand} \text{ or } rand_{i,j}[0,1] \leq CR_i^{(g)}, \\ x_{i,j}^{(g)} \text{ otherwise} \end{cases} \quad (4)$$

where j_{rand} is a random natural number in the range $[1, D]$. Similar to $F_i^{(g)}$, $CR_i^{(g)}$ is the globally adapted crossover rate associated with $x_i^{(g)}$. In the selection stage, fitness values of both $u_i^{(g)}$ and $x_i^{(g)}$ are compared and the one with superior fitness moves to the next generation.

D. Step reduction in population size

In second phase of the proposed algorithm, i.e., after about 60% function evaluations, we drastically reduce the population size from Np_{init} to Np_{intm} (intermediate population size) and perform the DE operations on them. The top Np_{intm} members, selected according to their fitness values after completion of the first phase of the algorithm, are involved during this stage with the main objective of exploitation. The size of Np_{intm} as $18 * D$, like in LSHADE algorithm, is found to be well-tuned for the operations of the algorithm.

E. Evolution during second phase

The population members which survive after the first phase undergo evolution in this phase. We employ the 'current-to-*p*best/1' mutation strategy of LSHADE algorithm as below:

$$v_i^{(g)} = x_i^{(g)} + F_i^{(g)}.\left(x_{pbest}^{(g)} - x_i^{(g)}\right) + F_i^{(g)}.\left(x_{R_1^i}^{(g)} - x_{R_2^i}^{(g)}\right) \quad (5)$$

Indices R_1^i & R_2^i are mutually exclusive, selected randomly from the range $[1, Np^{(g)}]$ where $Np^{(g)}$ denotes the dynamicity of population size in this phase. The population size is linearly reduced in second stage of the algorithm. $x_{pbest}^{(g)}$ refers to the best $Np^{(g)} \times p$ $(p \in [0,1])$ individuals of the current generation. We repair an element of the mutant vector as per Eq. (6) if the same goes outside the search boundary. The mutation operation is followed by the crossover operation which is same as in first phase of the algorithm.

$$v_{i,j}^{(g)} = \begin{cases} \dfrac{x_{min,j} + x_{i,j}^{(g)}}{2} \text{ if } v_{i,j}^{(g)} < x_{min,j} \\ \dfrac{x_{max,j} + x_{i,j}^{(g)}}{2} \text{ if } v_{i,j}^{(g)} > x_{max,j} \end{cases} \quad (6)$$

F. Control parameter adaptation

The control parameters, scale factor ($F \in [0,1]$) and crossover rate ($CR \in [0,1]$), are adapted in both stages of the algorithm

in a similar manner. Like LSHADE, we maintain H entries of historical memories, named as μF and μCR, for F and CR, respectively. Initially, all H entries of μF and μCR are set to 0.5. Subsequently, $F_i^{(g)}$ and $CR_i^{(g)}$, used by $x_i^{(g)}$ in g-th generation are generated by selecting an index r_i randomly within $[1, H]$ and performing the following operations:

$$F_i^{(g)} = randc\left(\mu F_{ri}^{(g)}, 0.1\right) \tag{7}$$

$$CR_i^{(g)} = randn\left(\mu CR_{ri}^{(g)}, 0.1\right) \tag{8}$$

where $randc\left(\mu F_{ri}^{(g)}, 0.1\right)$ is sampled from the Cauchy distribution with location parameter $\mu F_{ri}^{(g)}$ and scale parameter 0.1. $randn\left(\mu CR_{ri}^{(g)}, 0.1\right)$ is the normal distribution with mean $\mu CR_{ri}^{(g)}$ and variance 0.1. If $F_i > 1$, it is set to 1. For cases where $F_i \leq 0$, the same is regenerated. For $CR_i \notin [0,1]$, it is replaced with either 0 or 1, whichever is nearer.

The historical memories μF and μCR are updated based on the success history of past generations. In a generation g, if F and CR values can generate better trial vector $u_i^{(g)}$ than the respective target vector $x_i^{(g)}$, the values are stored in S_F and S_{CR}, respectively. At generation $g+1$, an entry at index k of the historical memory will be updated. Index k starts with 1 and incremented by 1 when a memory update takes place. If k exceeds the size of H, it is reset to 1. Suppose $\mu F_k^{(g)}$ and $\mu CR_k^{(g)}$ are the entries in the memory at index k in generation g. These will be updated in generation $g+1$ using weighted Lehmer mean (say, $mean_{WL}$) as follows [4-5]:

$$\mu F_k^{(g+1)} = mean_{WL}(S_F) \tag{9}$$

$$\mu CR_k^{(g+1)} = mean_{WL}(S_{CR}) \tag{10}$$

where $mean_{WL}(S)$ is defined as (S refers to either S_F or S_{CR}):

$$mean_{WL}(S) = \frac{\sum_{i=1}^{|S|} w_i . S_i^2}{\sum_{i=1}^{|S|} w_i . S_i} \tag{11}$$

$$w_i = \frac{\Delta f_i}{\sum_{j=1}^{|S|} \Delta f_j} \tag{12}$$

$$\text{and, } \Delta f_i = \left| f\left(u_i^{(g)}\right) - f(x_i^{(g)}) \right| \tag{13}$$

Eq. (13), which signifies the improvement in fitness in current generation, is used to influence the adaptation of the parameters.

G. Linear population size reduction (LPSR)

We reduce the population size linearly from Np_{intm} (intermediate population size) to NP_{min} (minimum population size) over the generations in second phase of the algorithm. Mathematically, the population size at generation $g+1$ is represented as [5]:

Algorithm 2: Steps in O-LSHADE

1. Dimension of the problem, D
2. Range of decision variables $[x_{min}, x_{max}]$
3. Define parameters for orthogonal array. Level Q and an integer J
4. Construct orthogonal array of N factors using Algorithm 1. Remove last $(N-D)$ columns from the array
5. Scale the elements in array columns as per the range $[x_{min}, x_{max}]$ to generate initial population of size Np_{init}
6. Compute Euclidean distance between each pair of members in Np_{init}. Define neighborhood of each member x_i as its nearest n_b individuals
7. Set generation counter $g = 0$, evaluation counter $NFE^{(g)} = 1$ and initialize control parameters μF and μCR. Specify historical memory size H of control parameters and maximum number of allowable function evaluations NFE_{max}

Loop (algorithm phase 1)

8. **while** termination criteria $NFE^{(g)} < round(0.6 * NFE_{max})$ **do**
9. **for** i = 1 to Np_{init} **do**
10. Adapt control parameters $F_i^{(g)}$ and $CR_i^{(g)}$ as per Eqs. (7)-(13).
11. Identify the neighborhood for $x_i^{(g)}$
12. Perform mutation within neighborhood to generate vector $v_i^{(g)}$ as per Eq. (2). Repair if necessary using Eq. (3)
13. Perform crossover to generate element $u_{i,j}^{(g)}$ as per Eq. (4). Formulate trial (offspring) vector $u_i^{(g)}$
14. Evaluate $f\left(u_i^{(g)}\right)$. Increase evaluation counter $NFE^{(g)}$
15. Select best fit individuals for next generation. If, $f\left(\mathrm{u}_i{}^{(g)}\right) \leq f\left(x_i^{(g)}\right)$, $x_i^{(g+1)} = u_i^{(g)}$, else $x_i^{(g+1)} = x_i^{(g)}$. End **for** loop.
16. Increase generation counter g. End **while** loop.
17. Sort all population members x_i for $i \in [0, Np_{init}]$ according to their fitness. Keep only top $18 * D$ members. Population size for second phase $Np_{intm} = 18 * D$
18. Calculate $NFE_{rem,max} = NFE_{max} - round(0.6 * NFE_{max})$. Initialize μF and μCR

Loop (algorithm phase 2)

19. **while** termination criteria $NFE^{(g)} < NFE_{max}$ **do**
20. **for** i = 1 to Np_{intm} **do**
21. Adapt control parameters $F_i^{(g)}$ and $CR_i^{(g)}$ as per Eqs. (7)-(13).
22. Perform mutation to generate vector $v_i^{(g)}$ as per Eq. (5). Repair if necessary using Eq. (6)
23. Perform crossover to generate element $u_{i,j}^{(g)}$ as per Eq. (4). Formulate trial (offspring) vector $u_i^{(g)}$
24. Evaluate $f\left(u_i^{(g)}\right)$. Increase evaluation counter $NFE^{(g)}$
25. Select best fit individuals for next generation. If, $f\left(\mathrm{u}_i{}^{(g)}\right) \leq f\left(x_i^{(g)}\right)$, $x_i^{(g+1)} = u_i^{(g)}$, else $x_i^{(g+1)} = x_i^{(g)}$. End **for** loop.
26. Update population size for next generation $Np^{(g+1)}$ as per LPSR strategy in Eq. (14).
27. Increase generation counter g. End **while** loop.

$$Np^{(g+1)} = round\left[Np_{intm} - \left(\frac{Np_{intm} - Np_{min}}{NFE_{rem,max}}\right) . NFE^{(g)}\right] \tag{14}$$

where $NFE^{(g)}$ is the current number of function evaluations at generation g, $NFE_{rem,max}$ is the maximum number of function evaluations remained after completion of the first phase (i.e., about 40% of the total allowable function evaluations). The mutation strategy needs minimum 4 individuals, hence $NP_{min} = 4$. When $Np^{(g+1)} < Np^{(g)}$, $\left[Np^{(g)} - Np^{(g+1)}\right]$ numbers of lowest ranked individuals (in terms of fitness values) are removed from the population.

The steps involved in the proposed O-LSHADE algorithm are summarized in Algorithm 2.

III. Experimental Results

The proposed O-LSHADE algorithm is tested on a variety of benchmark problems proposed in CEC2020 competition [23]. In the set of the competition problems, 1-unimodal function (F1), 3-basic functions (F2-F4), 3-hybrid functions (F5-F7), and 3-composite functions (F8-F10) are presented for assessment of an algorithm. These functions are required to be tested on 5, 10, 15, and 20-dimensional search space over 30 runs to account for the stochastic nature of an evolutionary algorithm.

A. Settings of algorithm parameters

The user defined input parameters for the algorithm are listed in Table I. These parameters, especially the initial population sizes, are selected after several trials. The increment in initial population size is roughly linear to the increase in problem dimension. Neighborhood size is kept small as the intention of the algorithm is to explore the space in the vicinity during first phase. The approach helps to enhance consistency in finding the optimum solutions. The settings of parameters for the second phase of O-LSHADE algorithm are in line with the LSHADE algorithm.

TABLE I. O-LSHADE Algorithm Parameter Settings

Parameter	Description	Settings	Note
Q	Orthogonal array level	30 for 5D 50 for 10D 60 for 15D 70 for 20D	
J	A positive integer	2	For all dimensions
NP_{init}	Initial population size	900 for 5D 2500 for 10D 3600 for 15D 4900 for 20D	Calculated as Q^J
n_b	Neighborhood size	6	
NFE_{max}	Maximum allowable function evaluations	50K for 5D 1M for 10D 3M for 15D 10M for 20D	Ref [23]. M stands for million
$NFE_{rem,max}$	Number of function evaluations remaining for second phase	$NFE_{max} - round(0.6 * NFE_{max})$	
Np_{intm}	Intermediate population size	18*D (best members)	Evaluated during second phase
H	Memory size for F and CR	5	For both phases

B. Statistical summary of results

The statistical summary of results over 30 independent runs of the algorithm for problems of all dimensions is presented in this section. The summary tables can be identified as - Table II for 5D, Table III for 10D, Table IV for 15D and Table V for 20D benchmark problems. Each table shows record of best fitness, worst fitness, fitness of median run, average fitness and standard deviation of fitness values over all 30 runs. The algorithm finds the optimal point for the unimodal function F1 in all dimensions over all runs as observed from the tabulated data. In the lowest dimensional search space (i.e., 5D), our proposed algorithm can attain the global optimum at least one time for all functions except function F3. However, the fitness attained for F3 in 5D is very close to 0.

TABLE II. Statistical Summary of 5-D Benchmark Problems

Func.	Best	Worst	Median	Mean	Std. dev.
F1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F2	0.00E+00	1.31E+02	1.30E-01	1.88E+01	4.16E+01
F3	4.97E-04	5.41E+00	2.34E-03	1.41E+00	1.36E+00
F4	0.00E+00	1.63E-01	1.09E-01	7.27E-02	5.98E-02
F5	0.00E+00	6.24E-01	0.00E+00	4.16E-02	1.58E-01
F6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F7	-	-	-	-	-
F8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F10	0.00E+00	3.00E+02	0.00E+00	1.13E+02	9.73E+01

TABLE III. Statistical Summary of 10-D Benchmark Problems

Func.	Best	Worst	Median	Mean	Std. dev.
F1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F2	1.18E+01	4.69E+01	2.70E+01	2.92E+01	1.04E+01
F3	3.91E+00	1.33E+01	4.26E+00	8.42E+00	3.23E+00
F4	0.00E+00	3.25E-01	1.62E-01	8.74E-02	7.70E-02
F5	0.00E+00	9.85E+00	0.00E+00	1.08E+00	1.91E+00
F6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F7	2.28E-06	8.12E-01	4.62E-05	2.29E-01	2.63E-01
F8	0.00E+00	1.01E+02	3.82E+01	3.36E+01	2.06E+01
F9	1.00E+02	1.00E+02	1.00E+02	1.00E+02	0.00E+00
F10	1.00E+02	1.00E+02	1.00E+02	1.00E+02	0.00E+00

TABLE IV. Statistical Summary of 15-D Benchmark Problems

Func.	Best	Worst	Median	Mean	Std. dev.
F1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F2	2.44E+00	2.81E+02	2.51E+02	1.10E+02	9.21E+01
F3	3.59E+00	2.83E+01	4.86E+00	1.23E+01	6.03E+00
F4	1.46E-01	7.71E-01	5.26E-01	5.37E-01	1.42E-01
F5	3.12E-01	1.49E+02	1.15E+00	1.56E+01	4.06E+01
F6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F7	1.13E-01	9.16E-01	6.90E-01	6.72E-01	1.82E-01
F8	1.00E+02	1.00E+02	1.00E+02	1.00E+02	0.00E+00
F9	1.00E+02	1.00E+02	1.00E+02	1.00E+02	0.00E+00
F10	1.00E+02	2.00E+02	2.00E+02	1.70E+02	4.66E+01

The increase in dimensionality of the search space escalates the challenges involved in finding the global optimum for multimodal, non-separable and sometimes asymmetrical functions. Our proposed algorithm is unable to locate the global optimum for basic functions F2 and F3 in search space of 10D or larger. Hybrid functions are formulated by randomly dividing the variables into subcomponents and applying different basic functions in different subcomponents. Hybrid functions are supposedly more complex than the basic functions. However, appropriate tuning of parameters in our algorithm helps to attain the global optimum at least once for functions F4 to F6 in 10D. Function F7 is hybridization of 5 basic functions, which makes

it difficult to find the optimal point. In general, our algorithm performs reasonably well on the hybrid functions in all dimensions.

TABLE V. STATISTICAL SUMMARY OF 20-D BENCHMARK PROBLEMS

Func.	Best	Worst	Median	Mean	Std. dev.
F1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F2	1.80E+00	3.45E+02	3.45E+00	1.15E+02	7.62E+01
F3	2.44E+00	4.60E+01	2.04E+01	2.52E+01	7.63E+00
F4	3.95E-01	4.41E+00	6.13E-01	1.01E+00	1.22E+00
F5	2.08E-01	1.27E+02	3.12E-01	1.78E+01	4.14E+01
F6	5.17E-01	5.17E-01	5.17E-01	5.17E-01	0.00E+00
F7	5.19E-01	1.14E+00	5.19E-01	8.42E-01	1.61E-01
F8	1.00E+02	1.03E+02	1.00E+02	1.00E+02	7.01E-01
F9	1.00E+02	2.01E+02	1.00E+02	1.10E+02	3.06E+01
F10	3.99E+02	4.25E+02	4.14E+02	4.10E+02	6.15E+00

Composite functions are the most difficult functions among the types included in CEC competitions. These functions are constructed merging the properties of their sub-functions so that continuity around the local/global optima is maintained. They use shifted and rotated basic functions, as well as some hybrid functions. The proposed algorithm O-LSHADE shows superior and consistent performance on composite problems of the lowest dimension 5D. For 10D, the algorithm can find the optimal point at least once for function F8. The algorithm converges to a local optimum for composite problems with higher dimensions or the problems consisting of four or more basic functions.

C. Algorithm complexity

The coding of the algorithm has been done using software **MATLAB 2015b** and the same is run on a personal computer with processor **Intel(R) Xeon(R) CPU E5-1620@3.50GHz and 16GB RAM**. The time durations (T_0, T_1 and $\widehat{T_2}$ in seconds) related to the algorithm are calculated as per the guidelines in [23] and provided in Table VI. It may be noted that a large part of the CPU run time is expended in creating the orthogonal array and thereafter calculating the distance between each pair of individuals. However, the population pool based on orthogonal design is fixed for all problems of a specific dimension. To elaborate specifically for 10D, 2500 population members ($NP_{init} = 2500$) are generated once and the associated distance matrix (of size 2500×2500) is also constructed once for all problems (here 10 nos.) and runs (here 30 runs for each problem). Therefore, once the above tasks are completed, the algorithm can be executed very fast for several runs as it is to complete only the required number of function evaluations.

TABLE VI. RUN TIME COMPLEXITY OF THE ALGORITHM

Dim	$T_0(s)$	$T_1(s)$	$\widehat{T_2}(s)$	$(\widehat{T_2} - T_1)/T_0$
D = 5	0.1041	0.4412	6.5551	58.7310
D = 10		0.4621	6.7140	60.0567
D = 15		0.4925	7.4485	66.8204

IV. CONCLUSION

Differential Evolution (DE) algorithm, its advanced version LSHADE and many other variants have shown remarkable performance in solving challenging optimization tasks including the numerical benchmark problems. In this paper, we present an algorithm, named as O-LSHADE, where orthogonal array-based initialization is incorporated in LSHADE algorithm. The initialization process ensures enough coverage and uniformity of the generated individuals over the search space. Furthermore, DE operations among the members of a small neighborhood at early stage explore the search space effectively and efficiently. We test the performance of O-LSHADE algorithm on all complex benchmark problems proposed for CEC2020 competition. The results prove the competitiveness of the proposed method. In future, we plan to compare the performance of our algorithm with several other methods. We also intend to try out few more operations on the population members generated with orthogonal array-based design.

REFERENCES

[1] Storn, Rainer, and Kenneth Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces." *Journal of global optimization* 11, no. 4 (1997): 341-359.

[2] Qin, A. Kai, Vicky Ling Huang, and Ponnuthurai N. Suganthan. "Differential evolution algorithm with strategy adaptation for global numerical optimization." *IEEE transactions on Evolutionary Computation* 13, no. 2 (2008): 398-417.

[3] Zhang, Jingqiao, and Arthur C. Sanderson. "JADE: adaptive differential evolution with optional external archive." *IEEE Transactions on evolutionary computation* 13, no. 5 (2009): 945-958.

[4] Tanabe, Ryoji, and Alex Fukunaga. "Success-history based parameter adaptation for differential evolution." In *2013 IEEE congress on evolutionary computation*, pp. 71-78. IEEE, 2013.

[5] Tanabe, Ryoji, and Alex S. Fukunaga. "Improving the search performance of SHADE using linear population size reduction." In *2014 IEEE congress on evolutionary computation (CEC)*, pp. 1658-1665. IEEE, 2014.

[6] Zhao, Shi-Zheng, Ponnuthurai N. Suganthan, and Swagatam Das. "Self-adaptive differential evolution with multi-trajectory search for large-scale optimization." *Soft Computing* 15, no. 11 (2011): 2175-2185.

[7] Wang, Yong, Zixing Cai, and Qingfu Zhang. "Differential evolution with composite trial vector generation strategies and control parameters." *IEEE Transactions on Evolutionary Computation* 15, no. 1 (2011): 55-66.

[8] https://www.ntu.edu.sg/home/epnsugan/

[9] Biswas, Partha P., Ponnuthurai N. Suganthan, and Gehan AJ Amaratunga. "Optimal placement of wind turbines in a windfarm using L-SHADE algorithm." In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 83-88. IEEE, 2017.

[10] Biswas, Partha P., Ponnuthurai N. Suganthan, and Gehan AJ Amaratunga. "Minimizing harmonic distortion in power system with optimal design of hybrid active power filter using differential evolution." *Applied Soft Computing* 61 (2017): 486-496.

[11] Biswas, Partha P., Noor H. Awad, Ponnuthurai N. Suganthan, Mostafa Z. Ali, and Gehan AJ Amaratunga. "Minimizing THD of multilevel inverters with optimal values of DC voltages and switching angles using LSHADE-EpSin algorithm." In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 77-82. IEEE, 2017.

[12] Biswas, Partha P., Rammohan Mallipeddi, Ponnuthurai N. Suganthan, and Gehan AJ Amaratunga. "Optimal reconfiguration and distributed generator allocation in distribution network using an advanced adaptive differential evolution." In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1-7. IEEE, 2017.

[13] Awad, Noor H., Mostafa Z. Ali, Ponnuthurai N. Suganthan, and R. G. Reynolds. "An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems." In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pp. 2958-2965. IEEE, 2016.

[14] Awad, Noor H., Mostafa Z. Ali, and Ponnuthurai N. Suganthan. "Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems." In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 372-379. IEEE, 2017.

[15] Brest, Janez, Mirjam Sepesy Maučec, and Borko Bošković. "iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization." In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1188-1195. IEEE, 2016.

[16] Brest, Janez, Mirjam Sepesy Maučec, and Borko Bošković. "Single objective real-parameter optimization: Algorithm jSO." In *2017 IEEE congress on evolutionary computation (CEC)*, pp. 1311-1318. IEEE, 2017.

[17] Mohamed, Ali W., Anas A. Hadi, Anas M. Fattouh, and Kamal M. Jambi. "LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems." In *2017 IEEE Congress on evolutionary computation (CEC)*, pp. 145-152. IEEE, 2017.

[18] Stanovov, Vladimir, Shakhnaz Akhmedova, and Eugene Semenkin. "LSHADE Algorithm with Rank-Based Selective Pressure Strategy for Solving CEC 2017 Benchmark Problems." In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1-8. IEEE, 2018.

[19] Viktorin, Adam, Roman Senkerik, Michal Pluhacek, Tomas Kadavy, and Ales Zamuda. "Distance based parameter adaptation for success-history based differential evolution." *Swarm and Evolutionary Computation* 50 (2019): 100462.

[20] Piotrowski, Adam P., and Jaroslaw J. Napiorkowski. "Step-by-step improvement of JADE and SHADE-based algorithms: Success or failure?." *Swarm and evolutionary computation* 43 (2018): 88-108.

[21] Mohamed, Ali W., Anas A. Hadi, and Kamal M. Jambi. "Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization." *Swarm and Evolutionary Computation* 50 (2019): 100455.

[22] Leung, Yiu-Wing, and Yuping Wang. "An orthogonal genetic algorithm with quantization for global numerical optimization." *IEEE Transactions on Evolutionary computation* 5, no. 1 (2001): 41-53.

[23] Yue, C.T., Kenneth Price, Ponnuthurai N. Suganthan, J. J. Liang, Mostafa Z. Ali, B. Y. Qu, Noor H. Awad, and Partha P. Biswas. "Problem definitions and evaluation criteria for the CEC 2020 special session and competition on single objective bound constrained numerical optimization." *Zhengzhou University and Nanyang Technological University, Zhenzhou, China, and Singapore, Tech. Rep* 201911 (2019).