Ensemble Sinusoidal Differential Covariance Matrix Adaptation with Euclidean Neighborhood for Solving CEC2017 Benchmark Problems

Noor H. Awad[1], Mostafa Z. Ali[2], Ponnuthurai N. Suganthan[1]
[1]Nanyang Technological University, Singapore 639798
School of Electrical & Electronic Engineering
[2]Jordan University of Science & Technology, Jordan 22110
School of Computer Information Systems
Emails: noor0029@e.ntu.edu.sg, mzali.pn@gmail.com, epnsugan@ntu.edu.sg

***Abstract*— Many Differential Evolution algorithms are introduced in the literature to solve optimization problems with diverse set of characteristics. In this paper, we propose an extension of the previously published paper LSHADE-EpSin that was ranked as the joint winner in the real-parameter single objective optimization competition, CEC 2016. The contribution of this work constitutes two major modifications that have been added to enhance the performance: ensemble of sinusoidal approaches based on performance adaptation and covariance matrix learning for the crossover operator. Two sinusoidal waves have been used to adapt the scaling factor: non-adaptive sinusoidal decreasing adjustment and an adaptive sinusoidal increasing adjustment. Instead of choosing one of the sinusoidal waves randomly, a performance adaptation scheme based on earlier success is used in this work. Moreover, covariance matrix learning with Euclidean neighborhood is used for the crossover operator to establish a suitable coordinate system, and to enhance the capability of LSHADE-EpSin to tackle problems with high correlation between the variables. The proposed algorithm, namely LSHADE-*cn*EpSin, is tested on the IEEE CEC2017 problems used in the Special Session and Competitions on Single Objective Bound Constrained Real-Parameter Single Objective Optimization. The results statistically affirm the efficiency of the proposed approach to obtain better results compared to other state-of-the-art algorithms.**

Keywords*— *Differential Evolution; Sinusoidal Wave, Ensemble approach; Covariance Matrix Learning.

I. INTRODUCTION

Differential Evolution (DE), firstly proposed by Storn and Price [1], has shown effective performance when solving different optimization problems with different characteristics [2-4]. DE simulates the natural evolution process by performing three main stages on a population space: mutation, crossover and selection. The population space is represented by a set of individuals which are uniformly distributed within the search range of the problem being solved. Trial vectors are generated using mutation and crossover operators, and then they are compared with the target vectors to select the best individuals which survived to the next generation. The aforementioned steps will be repeated until the termination criterion is met and the best solution is recorded. Many studies have proven the efficiency of applying DE to solve different optimization problems with diverse characteristics, including multimodality, non-separability, ill-conditioning and high dimensionality [5-7].

Differential Evolution is known to be highly dependent on the choice of parameter settings for its control parameters [7]. Based on this fact, many DE algorithms have been proposed in the literature to address this issue and to enhance the DE performance using some effective settings. Three control parameters are involved in DE which are: the scaling factor (F), the crossover rate (CR) and the population size (NP). Many studies have been focused on the development of adaptive or self-adaptive approaches aiming at adjusting the control parameter settings automatically [8-10].

CEC benchmark problems are among the widely used benchmarks which have attracted many researchers to use for testing their developed algorithms [5-7, 11]. Many DE algorithms have continuously been proposed to successfully solve these problems, and that is an indication of the usefulness of using DE-based algorithms to solve different optimization problems. LSHADE algorithm was ranked the winner of the single objective real parameter competition of the CEC'14 and demonstrated a very good performance when solving such benchmarks [12]. This algorithm is an extension of the SHADE algorithm which participated in the CEC 2013 competition and was ranked the third in real-parameter optimization case competition of that year [13]. Those two algorithms used JADE [10] algorithm adaptations in their bases. In JADE, the new mutation strategy "current/to/pbest" was proposed in which the authors used one of the top p best individuals to guide the evolutionary search. Moreover, JADE proposed an adaptive control setting for F and CR using Cauchy and Normal distributions respectively by utilizing previously successful mean values. On the other hand, SHADE enhances the performance of JADE by introducing a history-based scheme which selects a random index r_i from a memory M *of size* H in [1, H] to choose the mean values μ_F and μ_{CR}. LSHADE adds

978-1-5090-4601-0/17/$31.00 ©2017 IEEE

the population size reduction to linearly reduce the population size from N^{init} at each generation down to reach N^{min} at the end of the run.

A closer look at the literature reveals that there are a few studies for adapting the crossover operator of the differential evolution algorithm [14, 15]. The idea of using covariance matrix learning in DE was suggested in [14]. In that work, the authors adjusted the crossover operator of DE using a suitable coordinate system that was obtained by suitably adjusting the covariance matrix. In our proposed algorithm, the Euclidean neighborhood is used along with covariance matrix learning to establish a better coordinate system for the crossover operator in the proposed DE framework. The proposed work uses LSHADE-EpSin algorithm [16] which was ranked the joint winner with UMOEAsII [17] in the single parameter-operator competition. LSHADE-EpSin was introduced to enhance the performance of L-SHADE algorithm using an adaptive approach based on an ensemble of sinusoidal formulas to adapt the scaling factor (F) in an effective manner. In the basic algorithm of LSHADE-EpSin, one of the following two sinusoidal formulas is selected randomly to adapt F in the first half of the generations: non-adaptive sinusoidal decreasing adjustment and an adaptive sinusoidal increasing adjustment. In this paper, LSHADE-EpSin is enhanced by incorporating a performance adaptation scheme to effectively select one of the two sinusoidal formulas. Furthermore, the covariance matrix learning with Euclidean neighborhood is used to establish a proper coordinate system for selecting values for the crossover operator, in a way that suites the search mode.

The remainder of the paper is organized as follows. The proposed LSHADE-*cn*EpSin is presented in Section 2. The experimental set-up and simulation results are presented in Section 3. Finally, section 4 summarizes the conclusions of this work.

II. PROPOSED ALGORITHM

In this section, the proposed algorithm, LSHADE-*cn*EpSin, is explained in details. The algorithm starts by initializing a set of NP individuals within the search range of the problem being solved as shown below:

$$x^{j}_{i,0} = x^{j}_{\min} + rand(0,1).(x^{j}_{\max} - x^{j}_{\min}) \quad j = 1, 2, ..., D \qquad (1)$$

where j is the index of parameter value of the i^{th} individual vector at generation g=0, $rand(0,1)$ is a uniformly distributed random generator in the range [0,1] and $X_{\min} = \{x^{1}_{\min}, ..., x^{D}_{\min}\}, X_{\max} = \{x^{1}_{\max}, ..., x^{D}_{\max}\}$ are the lower and upper bounds of each decision variable x^{j}_{i}.

After that, the JADE mutation, "current/to/*p*best" is used to generate new trial vectors in each generation. In this mutation, as shown in Eq. 2, the best individual, $X_{pbest,g}$ is chosen from the top $NP \times p$ ($p \in [0,1]$) best individuals of the g[th] generation.

$$V_{i,g} = X_{i,g} + F_{i,g}.(X_{pbest,g} - X_{i,g}) + F_{i,g}.(X_{r_1,g} - X_{r_2,g}) \qquad (2)$$

where $X_{r_1,g}$ is chosen randomly with an index r_1 selected from [1, NP] and $X_{r_2,g}$ is chosen from the union of the population and an external archive A which stores the inferior parents recently replaced by the offspring ones. Initially the archive is filled with the initial population, P_0. At each generation g, if the size of A exceeds the maximum size, NP, random individuals are eliminated to make space for the newly added individuals.

A linear population size reduction is also used in the proposed algorithm as was in LSHADE-EpSin. The population size, NP, is adjusted in each generation g as shown in the following equation:

$$NP(g+1) = Round[(\frac{NP_{\min} - NP_{\max}}{FEs_{\max}}).FEs + NP_{\max}] \qquad (3)$$

Where $NP_{\min}$ is set to 4 which is the the minimum number of individuals needed to perform "current/to/*p*best" mutation strategy. $NP_{\max}$ is the maximum initial size of the population, FEs is the current function evaluation and $FEs_{\max}$ is the maximum function evaluations.

In LSHADE-*cn*EpSin, an ensemble of two sinusoidal waves are used to adapt the scaling factor based on successful performance of previous generations. The covariance matrix learning is also used for the crossover operator based on the Euclidean neighborhood. The following sub-sections explains those operations in details.

A. Parameter Adaptation

In LSHADE-*cn*EpSin, an ensemble of parameter adaptation is used to adapt the scaling factor, F. The first configuration is non-adaptive sinusoidal decreasing adjustment as shown in Eq. 4 where $freq$ represents the frequency of the sinusoidal function and g is the current generation number. In this configuration, the $freq$ parameter is set to a fixed value.

$$F_{i,g} = \frac{1}{2} * \left(\sin(2\pi * freq * g + \pi) * \frac{G_{\max} - g}{G_{\max}} + 1 \right) \qquad (4)$$

In the second configuration, an adaptive sinusoidal increasing adjustment is used as shown in Eq. 5.

$$F_{i,g} = \frac{1}{2} * \left(\sin(2\pi * freq_i * g) * \frac{g}{G_{\max}} + 1 \right) \qquad (5)$$

In this configuration, $freq_i$ is an adaptive frequency adjusted using an adaptive scheme at each generation g using Cauchy distribution as shown below:

$$freq_i = randc(\mu freq_{r_i}, 0.1) \quad (6)$$

where $\mu freq_{r_i}$ is chosen randomly from an external memory M_{freq} which stores the successful mean frequencies for the previous generations in S_{freq}. A random index number, $r_i \in [1, H]$, is chosen at the end of generation g, and $\mu freq_{r_i}$ is modified using Lehmer mean as will be explained next.

Each half of the allowed number of generations will have a different setup using both Sinusoidal configurations. In the first half of generations, $g \le \frac{G_{\max}}{2}$ where G_{max} is the maximum number of generations, a mixture of the two sinusoidal configurations based on performance adaptation is used.

In the sinusoidal pool, one of the two configurations is chosen by learning from their previous experiences in previous generations. A learning period, *LP*, is used to record the number of successful trial vectors generated by each of the sinusoidal configurations that can enter next generation, denoted as $ns_{g,k}$. Moreover, the number of trial vectors that are discarded in the next generation is also recorded in $nf_{g,k}$. In this notation, *k* can be 1 to denote non-adaptive sinusoidal decreasing adjustment and 2 for adaptive sinusoidal increasing adjustment. In the first *LP* generations, the two sinusoidal configurations have an equal probability, p_k, and one of them is chosen randomly. After that, the probabilities of choosing sinusoidal configurations are updated at each subsequent generation as follows:

$$p_k = \frac{S_{k,g}}{\sum_{k=1}^{K} S_{k,g}} \quad (7)$$

$$S_{k,g} = \frac{\sum_{i=g-LP}^{g-1} ns_{k,i}}{\sum_{i=g-LP}^{g-1} ns_{k,i} + \sum_{i=g-LP}^{g-1} nf_{k,i}} + \varepsilon \quad (8)$$

Where $S_{k,g}$ represents the success rate of trial vectors generated by one of the sinusoidal configurations. To avoid the null success rates, ε is added and is set to 0.01.

For the second half of generations, $g > \frac{G_{\max}}{2}$, the scaling factor, $F_{i,g}$ is adapted using Cauchy distribution as shown below:

$$F_{i,g} = randc(\mu F_{r_i}, 0.1) \quad (9)$$

As in the original L-SHADE, the crossover rate $CR_{i,g}$ is adapted using normal distribution all over the generations:

$$CR_{i,g} = randn(\mu CR_{r_i}, 0.1) \quad (10)$$

where μF_{r_i}, μCR_{r_i} are chosen randomly from the successful means of previous generations which are stored in a memory *M*, and r_i is a random index within the memory size. Initially, μF, μCR are both set to 0.5 and are updated for the next generations. At the end of generation g, the memory is updated where index *k* determines the position in memory to be updated and is incremented when new values are stored in the memory. The two means μF, μCR are modified using the weighted Lehmer mean, $mean_{WL}$, and they are computed as shown in the following equations where U_i is the trial vector and X_i is the target vector:

$$\mu F_{k,g+1} = mean_{WL}(S_F) \quad (11)$$

$$\mu CR_{k,g+1} = mean_{WL}(S_{CR}) \quad (12)$$

$$mean_{WL}(S) = \frac{\sum_{k=1}^{|S|} w_k . S_k^2}{\sum_{k=1}^{|S|} w_k . S_k} \quad (13)$$

$$w_k = \frac{\Delta f_k}{\sum_{j=1}^{|S|} \Delta f_j} \quad (14)$$

$$\Delta f_k = | f(U_{k,G}) - f(X_{k,G}) | \quad (15)$$

B. Covariance Matrix Learning with Euclidean neighborhood

In this proposed algorithm, the crossover operator is performed based on the covariance matrix learning with Euclidean neighborhood with a probability *pc*. In this mechanism, the individuals are first sorted according to their function values and the best individual, X_{best}, is marked. Next, the Euclidean distance is computed between X_{best} and every other individual in the population. After that, the individuals are sorted according to their Euclidean distance and $NP \times ps$

individuals are chosen to form a neighborhood region around the best individual. *ps* is the proportion of individuals that are used to generate the covariance matrix, and since the population size is dynamically decreased then $NP \times ps$ is dynamically decreased also. The covariance matrix, *C* is then computed from this neighborhood region as shown below:

$$C = BDB^T \tag{16}$$

Where B^T and B are orthogonal matrices and *D* is the diagonal matrix which consists of the eigenvalues.

After the covariance matrix is computed, the target and trial vectors are updated using B^T as follows:

$$X'_{i,g} = B^T X_{i,g} \tag{17}$$

$$V'_{i,g} = B^T V_{i,g} \tag{18}$$

The binomial crossover is then applied on $X'_{i,g}$ and $V'_{i,g}$ to create a trial vector $U'_{i,g}$ in the Eigen coordinate system as follows:

$$u'_{i,j,g} = \begin{cases} v'_{i,j,g}, & \text{if rand}(0,1) \le CR_{i,g} \text{ or } j = j_{rand} \\ x'_{i,j,g}, & \text{otherwise} \end{cases} \tag{19}$$

Finally, $U'_{i,g}$ is transformed into the original coordinate system using B:

$$U_{i,g} = BU'_{i,g} \tag{20}$$

C. Putting it all together

The pseudo-code of LSHADE-*cn*EpSin algorithm is presented in Fig. 1. Lines 1-4 represent the initialization of the population and set-up of the initial values for the used parameters. Lines 6-19 describe the parameter adaptation for *F* and *CR* as discussed in Sub-Section A. On the other hand, lines 20-29 describe the generation of new trial vectors using "current-to-*p*best" and the covariance matrix learning with Euclidean neighborhood as explained in Section B. Next, the memory is updated and finally the linear population size reduction is computed in lines 30-35.

III. Experimental Results

A. Numerical benchmarks

The performance of the proposed algorithm, LSHADE-*cn*EpSin, is evaluated using a set of problems presented in the CEC2017 competition on single objective bound constrained real-parameter optimization. This benchmark contains 30 test functions with a diverse set of characteristics. The functions are tested on 10*D*, 30*D* and 50*D*. In summary, functions 1–3 are unimodal, functions 4–10 are multimodal, functions 11–20 are hybrid, and functions 21–30 are composition functions. More details can be found in [18].

Algorithm: LSHADE-*cn*EpSin

1. Initialize population at first generation g_0, $P_{g_0} = \langle x_1^{g_0}, \ldots, x_{NP}^{g_0} \rangle$
2. Initialize memory of first control settings *M:* μF and μCR with 0.5
3. Initialize memory of second control settings M_{freq}: $\mu Freq$ with 0.5
4. Initialize covariance matrix settings, *ps* = 0.5, *pc* = 0.4
5. **While** termination criterion is not met **Do**
6. **if** $g \le G_{\max} / 2$
7. **if** $g \le LP$
8. Both sinusoidal configurations have the same probability, p_1=p_2=0.5
9. **Else**
10. Calculate the two probabilities using Eqs. 7-8
11. **EndIf**
12. Adapt $F_{i,g}$ using the sinusoidal configuration with larger probability
13. **EndIf**
14. **if** $g > G_{\max} / 2$
15. Reset successful mean arrays: $S_F = \phi$
16. Generate a random index $r_i = rand(1, H)$
17. Generate $F_{i,g} = randc(\mu F_{r_i}, 0.1)$
18. **EndIf**
19. **Generate** *CRi* using $CR_{i,g} = randn(\mu CR_{r_i}, 0.1)$
20. **For** *i*=1to *NP*
21. Generate new mutant vector $V_{i,g}$ using Eq. 2
22. **If** *rand* < *pc*
23. Apply covariance matrix learning for crossover operator as explained in sub-section B.
24. **Else**
25. Apply original binomial crossover
26. **EndIf**
27. Store successful F_i and CR_i for both control parameter settings
28. **EndFor**
29. Update memory as shown in Eqs. 11-15
30. **Call** linear population size reduction
31. **Apply** Eq. 3 to calculate population size for next generation $NP(g+1)$
32. Calculate $NP_{diff} = NP(g) - NP(g+1)$
33. Sort individuals P_g based on functions values
34. Eliminate worst individuals NP_{diff} from P_g
35. **EndWhile**

Fig. 1. Pseudo-code of LSHADE-*cn*EpSin algorithm

B. Algorithm parameters

The parameter values of LSHADE-*cn*EpSin algorithm are set as follows:

- The initial values of all μF, μCR are both set to 0.5. For the first half of generations, $\mu freq$ and μCR are

stored in a two-dimensional memory. For the second half, the same two-dimensional memory is used in which $\mu freq$ is replaced with μF which is initially set to 0.5. The Memory size H is set to 5. For non-adaptive decreasing adjustment, $freq$ is set to 0.5.

- After conducting extensive experiments, ps is set to 0.5 and pc is set to 0.4. For space limitations, sensitivity analysis for those parameters are not included.
- $NP_{\max}$ and $NP_{\min}$ are set to $18 * D$ and 4, respectively.

C. Algorithm complexity

The algorithm complexity of LSHADE-*cn*EpSin is shown in Table I. The algorithm was coded using Matlab 2015b and was run on a PC with an Intel CPU (3.40GHz) and 8GB RAM. The computational complexity is calculated as described in [18]. In Table I, T_0 denotes the running time of the following program:

for $i = 1:1000000$

$x = 0.55 + (double)i; x = x + x; x = x / 2; x = x * x;$

$x = sqrt(x);\ \ x = \log(x); x = \exp(x); x = x / (x + 2);$

end

TABLE I. ALGORITHM COMPLEXITY

	T_0	T_1	$\hat{T}_2$	$(\hat{T}_2 - T_1) / T_0$
$D = 10$	0.1648	0.5408	3.7811	19.6654
$D = 30$		0.8582	3.4794	15.9080
$D = 50$		1.5500	3.7338	13.2531

T_1 is the computing time for f_{18} for 200,000 evaluations while T_2 is the complete running time of the proposed algorithm for f_{18}, with a total budget of 200,000 evaluations. T_2 is evaluated five times, and the mean for T_2 is denoted as $\hat{T}_2$. Finally, the algorithm complexity is shown as $\hat{T}_2, T_1$ and $(\hat{T}_2 - T_1) / T_0$.

D. Statistical results

The algorithm was run 51 times for each test problem with a number of function evaluations equals to $10,000 \times D$. When the difference between the best found solution and the optimal became less than 10^{-8}, the error was treated as 0. The statistical results of LSHADE-*cn*EpSin on D=10, 30, 50 and 100 are presented in Tables II-V respectively. Each table gives the best, worst, median, mean and the standard deviation over the 51 runs of the error value between the best fitness values found in each run and the true optimal value.

For unimodal functions, F_1-F_3, the algorithm was successfully able to obtain the optimal solution except for F_2 in 50D. Considering multimodal functions, F_4-F_{10}, the algorithm was able to obtain the optimal solution for F_9 for 10D, 30D and 50D. The same case for F_6 except for 50D in which the results were very close to the optimal.

In regards to hybrid functions, F_{11}-F_{20}, the algorithm was able to obtain good solutions except for F_{12} in which the performance was reduced when D increased. Considering the composition functions, F_{21}-F_{30}, the algorithms' performance in terms of mean values was worse than the other functions. It is noted that these functions were difficult as they contain huge number of local optimum.

TABLE II. STATISTICAL RESULTS OF THE 10-D BENCHMARK FUNCTIONS, AVERAGED OVER 51 INDEPENDENT RUNS

	Best	Worst	Median	Mean	Std
F1	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F2	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F3	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F4	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F5	0.0000E+00	2.9907E+00	1.9901E+00	1.6851E+00	7.5340E-01
F6	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F7	1.0631E+01	1.2900E+01	1.2020E+01	1.1980E+01	4.7993E-01
F8	1.6958E-03	2.9868E+00	1.9907E+00	1.7969E+00	7.7141E-01
F9	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F10	3.7069E-01	1.5505E+02	1.5102E+01	4.3025E+01	5.5742E+01
F11	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F12	2.0814E-01	2.3838E+02	1.1865E+02	1.0128E+02	7.3033E+01
F13	0.0000E+00	8.3169E+00	4.8371E+00	3.6570E+00	2.6566E+00
F14	0.0000E+00	9.9496E-01	0.0000E+00	7.8036E-02	2.7016E-01
F15	7.0343E-06	5.0000E-01	4.9114E-01	3.2389E-01	2.1622E-01
F16	3.9170E-03	1.0845E+00	4.9843E-01	5.3722E-01	2.9342E-01
F17	2.6571E-03	2.6340E+00	3.2271E-01	3.0723E-01	3.8145E-01
F18	2.2212E-04	2.0489E+01	4.6182E-01	3.8592E+00	7.6265E+00
F19	0.0000E+00	1.5000E+00	1.9729E-02	4.4653E-02	2.0877E-01
F20	0.0000E+00	6.2435E-01	3.1217E-01	2.5708E-01	2.3114E-01
F21	1.0000E+02	2.0406E+02	1.0000E+02	1.4636E+02	5.1667E+01
F22	1.0000E+02	1.0035E+02	1.0000E+02	1.0001E+02	6.8026E-02
F23	3.0000E+02	3.0455E+02	3.0270E+02	3.0200E+02	1.6424E+00
F24	1.0000E+02	3.3198E+02	3.2954E+02	3.1583E+02	5.4512E+01
F25	3.9774E+02	4.4337E+02	4.4333E+02	4.2556E+02	2.2359E+01
F26	3.0000E+02	3.0000E+02	3.0000E+02	3.0000E+02	0.0000E+00
F27	3.8424E+02	3.9461E+02	3.8863E+02	3.8950E+02	1.9636E+00
F28	3.0000E+02	6.1107E+02	3.0000E+02	3.8488E+02	1.1882E+02
F29	2.2603E+02	2.3315E+02	2.2824E+02	2.2841E+02	1.7219E+00
F30	3.3950E+02	4.6462E+05	4.0743E+02	1.7618E+04	8.6130E+04

TABLE III. STATISTICAL RESULTS OF THE 30-D BENCHMARK FUNCTIONS, AVERAGED OVER 51 INDEPENDENT RUNS

	Best	Worst	Median	Mean	Std
F1	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F2	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F3	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F4	3.4763E+01	5.0102E+01	4.2539E+01	4.2282E+01	3.0697E+00
F5	5.5543E+00	1.7449E+01	1.2464E+01	1.2251E+01	2.3430E+00
F6	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F7	3.8949E+01	4.8061E+01	4.3271E+01	4.3295E+01	2.1667E+00
F8	5.4402E+00	1.7350E+01	1.3428E+01	1.2926E+01	2.8641E+00
F9	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F10	8.4381E+02	1.7186E+03	1.4204E+03	1.3884E+03	2.1047E+02
F11	1.7249E+00	5.9012E+01	3.9632E+00	1.3539E+01	1.9384E+01
F12	1.3150E+02	9.3455E+02	3.4421E+02	3.7245E+02	2.0053E+02
F13	2.0256E+00	7.9630E+01	1.7147E+01	1.7258E+01	1.0207E+01
F14	7.6082E+00	2.4002E+01	2.2031E+01	2.1573E+01	2.2600E+00
F15	5.7530E-01	9.8202E+00	2.8551E+00	3.2400E+00	1.9802E+00
F16	3.9799E+00	1.4617E+02	1.4469E+01	2.2884E+01	3.0731E+01
F17	1.5817E+01	3.9496E+01	2.8845E+01	2.8601E+01	5.5593E+00
F18	2.0199E+01	2.3461E+01	2.1307E+01	2.1085E+01	7.5204E-01
F19	2.9633E+00	1.1938E+01	5.8400E+00	5.8283E+00	1.9247E+00
F20	1.3089E+01	5.3375E+01	3.0439E+01	3.0346E+01	7.3523E+00
F21	2.0719E+02	2.1637E+02	2.1225E+02	2.1205E+02	2.5616E+00
F22	1.0000E+02	1.0000E+02	1.0000E+02	1.0000E+02	1.0047E-13
F23	3.4488E+02	3.6528E+02	3.5574E+02	3.5615E+02	3.7319E+00
F24	4.1803E+02	4.3361E+02	4.2919E+02	4.2848E+02	2.9483E+00
F25	3.8666E+02	3.8671E+02	3.8667E+02	3.8668E+02	8.8975E-03
F26	8.4588E+02	1.0287E+03	9.5516E+02	9.4861E+02	4.6027E+01
F27	4.8815E+02	5.1642E+02	5.0500E+02	5.0416E+02	6.6996E+00
F28	3.0000E+02	4.1397E+02	3.0000E+02	3.1522E+02	3.8592E+01
F29	4.1850E+02	4.5345E+02	4.3495E+02	4.3456E+02	7.3625E+00
F30	1.9414E+03	2.1340E+03	1.9705E+03	1.9774E+03	4.1663E+01

TABLE IV. STATISTICAL RESULTS OF THE 50-*D* BENCHMARK FUNCTIONS, AVERAGED OVER 51 INDEPENDENT RUNS

	Best	Worst	Median	Mean	Std
F1	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F2	0.0000E+00	1.0000E+01	1.0000E+00	1.5686E+00	1.9314E+00
F3	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F4	7.8572E+00	1.4036E+02	2.6061E+01	5.1401E+01	4.4262E+01
F5	1.3272E+01	3.5606E+01	2.6405E+01	2.5166E+01	6.4447E+00
F6	4.7943E-08	6.1479E-06	5.8730E-07	9.1569E-07	1.0750E-06
F7	6.6948E+01	8.8001E+01	7.4775E+01	7.6639E+01	6.0618E+00
F8	1.2766E+01	3.6112E+01	2.8914E+01	2.6319E+01	6.5917E+00
F9	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F10	2.2505E+03	3.7451E+03	3.2468E+03	3.2001E+03	3.3972E+02
F11	1.6077E+01	2.5582E+01	2.1768E+01	2.1393E+01	2.0902E+00
F12	6.4344E+02	2.2630E+03	1.4302E+03	1.4753E+03	3.6472E+02
F13	9.2721E+00	1.2805E+02	5.1104E+01	6.9430E+01	3.4457E+01
F14	2.2765E+01	3.5666E+01	2.6313E+01	2.6522E+01	2.4924E+00
F15	2.0085E+01	3.9443E+01	2.4572E+01	2.5596E+01	4.0567E+00
F16	1.3807E+02	4.8091E+02	2.7377E+02	2.7453E+02	9.9692E+01
F17	7.1233E+01	3.8519E+02	2.3373E+02	2.0706E+02	7.3064E+01
F18	2.0499E+01	3.1574E+01	2.4243E+01	2.4332E+01	2.1179E+00
F19	1.2388E+01	2.3010E+01	1.7344E+01	1.7406E+01	2.4713E+00
F20	7.2420E+01	2.8840E+02	1.0854E+02	1.1412E+02	3.5483E+01
F21	2.1676E+02	2.4058E+02	2.2545E+02	2.2676E+02	7.0598E+00
F22	1.0000E+02	3.9457E+03	1.2506E+02	1.5950E+03	1.6659E+03
F23	4.2806E+02	4.5597E+02	4.4029E+02	4.3929E+02	6.9001E+00
F24	5.0407E+02	5.2279E+02	5.1175E+02	5.1282E+02	5.5948E+00
F25	4.8013E+02	4.8790E+02	4.8018E+02	4.8034E+02	1.0816E+00
F26	9.7920E+02	1.3883E+03	1.2375E+03	1.2026E+03	1.1870E+02
F27	5.0846E+02	5.5020E+02	5.2476E+02	5.2543E+02	9.2143E+00
F28	4.5287E+02	5.0676E+02	4.5637E+02	4.5913E+02	1.1904E+01
F29	3.3281E+02	3.8024E+02	3.5294E+02	3.5289E+02	9.7796E+00
F30	5.7773E+05	8.7439E+05	6.4612E+05	6.5753E+05	7.2413E+04

TABLE V. STATISTICAL RESULTS OF THE 100-*D* BENCHMARK FUNCTIONS, AVERAGED OVER 51 INDEPENDENT RUNS

	Best	Worst	Median	Mean	Std
F1	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F2	4.0000E+00	4.3981E+12	6.7020E+03	9.5737E+10	6.1800E+11
F3	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F4	1.7955E+02	2.0738E+02	1.9284E+02	1.9811E+02	8.2973E+00
F5	4.4628E+01	1.1382E+02	5.5080E+01	5.5877E+01	9.9060E+00
F6	2.3057E-05	1.3058E-04	5.5521E-05	6.0247E-05	2.1831E-05
F7	1.4223E+02	1.9256E+02	1.6171E+02	1.6225E+02	7.9055E+00
F8	4.2105E+01	6.7538E+01	5.3938E+01	5.3519E+01	5.3905E+00
F9	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F10	8.5005E+03	1.1298E+04	1.0365E+04	1.0295E+04	5.2072E+02
F11	2.1658E+01	1.1491E+02	3.4086E+01	4.9236E+01	3.0217E+01
F12	2.9277E+03	5.9983E+03	4.6293E+03	4.6184E+03	6.4825E+02
F13	5.1339E+01	2.2260E+02	1.2762E+02	1.2546E+02	3.6523E+01
F14	3.7150E+01	6.9086E+01	4.8014E+01	4.9746E+01	8.1671E+00
F15	4.6238E+01	1.5392E+02	8.3166E+01	8.9898E+01	2.8333E+01
F16	7.1241E+02	1.7299E+03	1.2311E+03	1.2223E+03	2.3592E+02
F17	5.8246E+02	1.2221E+03	9.4412E+02	9.3195E+02	1.7421E+02
F18	4.4642E+01	1.1819E+02	7.6278E+01	7.7869E+01	1.9946E+01
F19	4.2393E+01	6.9814E+01	5.5889E+01	5.5469E+01	6.0498E+00
F20	4.7534E+02	1.4516E+03	1.1204E+03	1.0772E+03	2.1561E+02
F21	2.6135E+02	2.9159E+02	2.7743E+02	2.7732E+02	6.9385E+00
F22	8.7167E+03	1.1326E+04	1.0540E+04	1.0425E+04	5.2996E+02
F23	5.7632E+02	6.1581E+02	5.9833E+02	5.9781E+02	7.6890E+00
F24	8.9890E+02	9.6386E+02	9.1415E+02	9.1684E+02	1.3440E+01
F25	5.7576E+02	7.7267E+02	6.9675E+02	6.8426E+02	4.3431E+01
F26	2.9239E+03	3.4077E+03	3.1099E+03	3.1110E+03	1.2204E+02
F27	5.5929E+02	6.1299E+02	5.8719E+02	5.8863E+02	1.3115E+01
F28	4.7715E+02	5.7060E+02	5.1624E+02	5.1543E+02	2.2015E+01
F29	8.9864E+02	1.4710E+03	1.1123E+03	1.1194E+03	1.4878E+02
F30	2.0831E+03	2.8202E+03	2.3506E+03	2.3583E+03	1.4367E+02

*E. Comparing LSHADE-*cn*EpSin with other state-of-the-art algorithms.*

In this subsection, the performance of the LSHADE-*cn*EpSin algorithm is compared with the performances of other state-of-the-art algorithms as listed below:

1. Self-adaptive differential evolution algorithm for numerical optimization (SaDE) [8].
2. Adaptive differential evolution with optional external archive (JADE) [10].
3. Differential Evolution with Success-History Based Parameter Adaptation (SHADE) [12].
4. Differential Evolution with Linear Population Size Reduction (L-SHADE) [13].
5. Testing United Multi operator Evolutionary Algorithms-II on Single Objective Optimization Problems (UMOEAsII) [17].
6. Evaluating the Mean-Variance Mapping Optimization on the IEEE-CEC 2014 Test Suite (MVMO) [19].

The same parameter values that were suggested in the original papers were used to run the tests when they were used in the comparison against the proposed work. Each algorithm was run for $10,000 \times D$ functions evaluations for each of 51 independent runs. For space limitation, the average mean values over 51 runs for 50*D* is reported as shown in Table VI. The best error value is marked in boldface. LSHADE-*cn*EpSin performed better than the winner of the 2014 competition (L-SHADE), and also better than UMOEAsII which ranked the joint winner in CEC 2016 competition. This affirms the effectiveness of the proposed algorithm and its control settings.

Moreover, the Wilcoxon's rank-sum test was used to judge the significance of the results [20]. The last row of Table VI summarizes the results of this test at 0.05 significance level comparing each algorithm versus LSHADE-*cn*EpSin on the 30 functions as (*w*/*t*/*l*) denoted as *w*(+:win)/*t*(=:tie)/*l*(–:lose). For each of the competitive functions in this table, "–" identifies that the performance of the contestant algorithm is worse than LSHADE-*cn*EpSin, "+" marks the instances where the proposed algorithm exhibits a superior performance, and "=" indicates that the performance difference between the competitor and LSHADE-*cn*EpSin is not statistically significant. As shown in this table, LSHADE-*cn*EpSin clearly has the best overall performance among all the variants and significantly improves upon the performance of L-SHADE and UMOEAsII algorithms. LSHADE performs better than LSHADE-*cn*EpSin in 7 functions, shows a similar performance in 9 functions, and shows an inferior performance in 14 functions. On the other hand, UMOEAsII performs better than LSHADE-*cn*EpSin in 8 functions, shows a similar performance in 4 functions, and shows an inferior performance in 18 functions.

IV. CONCLUSION

This paper introduces LSHADE-*cn*EpSin in which an ensemble of parameter adaptation for scaling factor is introduced. This ensemble approach consists of a mixture of two sinusoidal formulas: the non-adaptive sinusoidal decreasing adjustment, and the adaptive sinusoidal increasing adjustment. This mixture is activated in the first half of generations. The selection between those configurations is based on successful performances for previous generations

TABLE VI. MEAN AND STANDARD DEVIATION OF THE ERROR VALUES FOR FUNCTIONS F1-F30 AVERAGED OVER 51 RUNS @ 50D. BEST ENTRIES ARE MARKED IN BOLDFACE.

	SaDE	JADE	SHADE	LSHADE	UMOEAsII	MVMO	LSHADE-*cn*EpSin
F1	1.2109E+03 (1.9748E+03) –	5.2385E-14 (2.5180E-14) =	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	1.3313E-05 (5.6019E-06) –	**0.0000E+00 (0.0000E+00)**
F2	9.2784E+01 (4.1214E+01) –	1.3112E+13 (8.5354E+13) –	1.0801E+12 (4.3906E+12) –	4.1176E-01 (6.6862E-01) –	**0.0000E+00 (0.0000E+00)** =	1.8060E+17 (1.2778E+18) –	1.5686E+00 (1.9314E+00) –
F3	2.7155E+02 (8.2894E+02) –	1.7712E+04 (3.7017E+04) –	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	2.1202E-09 (8.8715E-09) –	5.3095E-07 (1.0965E-07) –	**0.0000E+00 (0.0000E+00)** –
F4	8.9258E+01 (4.2166E+01) –	4.9625E+01 (4.7914E+01) =	5.6885E+01 (4.6262E+01) –	8.1837E+01 (4.8372E+01) –	6.5462E+01 (5.2164E+01) –	**3.5808E+01 (3.6684E+01)** +	5.1401E+01 (4.4262E+01) +
F5	9.2316E+01 (1.8646E+01) –	5.4288E+01 (8.8034E+00) –	3.2859E+01 (5.0387E+00) –	1.2244E+01 (2.0482E+00) +	**5.0801E+00 (1.6684E+00)** +	8.0787E+01 (1.6432E+01) –	2.5166E+01 (6.4447E+00) –
F6	7.4347E-03 (2.3519E-02) –	**1.4489E-13 (9.1172E-14)** =	8.3876E-04 (1.0169E-03) –	5.6921E-05 (3.7147E-04) –	1.1951E-06 (1.9013E-06) –	5.4321E-03 (3.3038E-03) –	9.1569E-07 (1.0750E-06) –
F7	1.4098E+02 (1.9307E+01) –	1.0140E+02 (6.4883E+00) –	8.0964E+01 (3.7800E+00) –	6.3236E+01 (1.7083E+00) +	**5.6459E+01 (7.1546E-01)** +	1.2320E+02 (1.2795E+01) –	7.6639E+01 (6.0618E+00) –
F8	9.4209E+01 (1.7744E+01) –	5.5234E+01 (7.7643E+00) –	3.2355E+01 (3.8252E+00) –	1.1979E+01 (2.2789E+00) +	**4.7781E+00 (1.6264E+00)** +	7.5910E+01 (1.6122E+01) –	2.6319E+01 (6.5917E+00) –
F9	4.8300E+01 (6.2913E+01) –	1.1773E+00 (1.3141E+00) –	1.1123E+00 (9.3715E-01) –	**0.0000E+00 (0.0000E+00)** =	1.7555E-03 (1.2536E-02) –	7.3843E+00 (5.7735E+00) –	**0.0000E+00 (0.0000E+00)** –
F10	6.6053E+03 (1.6328E+03) –	3.7500E+03 (2.5448E+02) –	3.3444E+03 (2.9402E+02) –	**3.1792E+03 (2.5493E+02)** +	3.3804E+03 (4.7255E+02) –	3.4971E+03 (4.3138E+02) –	3.2001E+03 (3.3972E+02) –
F11	1.0935E+02 (3.5435E+01) –	1.3612E+02 (3.3972E+01) –	1.2065E+02 (2.9317E+01) –	4.8617E+01 (7.9192E+00) –	4.5701E+01 (9.1852E+00) –	4.7488E+01 (8.7237E+00) –	**2.1393E+01 (2.0902E+00)** –
F12	1.1159E+05 (6.2044E+04) –	5.1468E+03 (3.3233E+03) –	5.1362E+03 (2.8785E+03) –	2.1677E+03 (4.5156E+02) –	2.1449E+03 (5.3559E+02) –	1.2955E+03 (2.7935E+02) –	**1.4753E+03 (3.6472E+02)** –
F13	1.2015E+03 (1.4531E+03) –	3.0338E+02 (2.6999E+02) –	2.6565E+02 (1.4944E+02) –	6.2678E+01 (2.8315E+01) =	5.1787E+01 (2.1985E+01) +	**4.3776E+01 (1.7622E+01)** +	6.9430E+01 (3.4457E+01) +
F14	2.1804E+03 (2.2053E+03) –	1.0519E+04 (3.1138E+04) –	2.1578E+02 (7.2995E+01) –	2.9084E+01 (2.9230E+00) =	2.9299E+01 (2.4831E+00) =	4.8524E+01 (1.2153E+01) –	**2.6522E+01 (2.4924E+00)** –
F15	3.3579E+03 (2.7961E+03) –	3.4992E+02 (4.4266E+02) –	3.2262E+02 (1.4201E+02) –	4.0764E+01 (9.9154E+00) –	4.1468E+01 (1.0651E+01) –	4.4630E+01 (1.1280E+01) –	**2.5596E+01 (4.0567E+00)** –
F16	8.1704E+02 (2.3477E+02) –	8.5696E+02 (1.7532E+02) –	7.3389E+02 (1.8854E+02) –	3.7654E+02 (1.1760E+02) –	3.9288E+02 (1.5514E+02) –	8.4082E+02 (1.9349E+02) –	**2.7453E+02 (9.9692E+01)** –
F17	5.0845E+02 (1.5357E+02) –	6.0010E+02 (1.2128E+02) –	5.1634E+02 (1.1109E+02) –	2.5490E+02 (7.4552E+01) –	3.1356E+02 (1.0636E+02) –	5.1999E+02 (1.3382E+02) –	**2.0706E+02 (7.3064E+01)** –
F18	3.2436E+04 (1.6833E+04) –	1.8906E+02 (1.2561E+02) –	1.8946E+02 (1.0338E+02) –	3.9292E+01 (1.1076E+01) –	3.5997E+01 (8.7118E+00) –	4.1756E+01 (1.9445E+01) –	**2.4332E+01 (2.1179E+00)** –
F19	1.1397E+04 (4.9099E+03) –	3.2429E+02 (1.2561E+03) –	1.5976E+02 (5.6842E+01) –	2.4584E+01 (8.8157E+00) –	2.2807E+01 (3.7669E+00) –	**1.7338E+01 (5.1321E+00)** =	**1.7406E+01 (2.4713E+00)** –
F20	3.5284E+02 (1.5090E+02) –	4.3806E+02 (1.3382E+02) –	3.3382E+02 (1.2079E+02) –	1.7397E+02 (7.9274E+01) –	2.3041E+02 (1.2312E+02) –	3.2965E+02 (1.4772E+02) –	**1.1412E+02 (3.5483E+01)** –
F21	2.8717E+02 (1.3646E+01) –	2.5198E+02 (9.6384E+00) –	2.3338E+02 (5.1139E+00) –	2.1272E+02 (1.9463E+00) +	**2.0681E+02 (2.5498E+00)** +	2.7719E+02 (1.6036E+01) –	2.2676E+02 (7.0598E+00) –
F22	2.9282E+03 (3.2427E+03) –	3.3364E+03 (1.8053E+03) –	3.1774E+03 (1.5566E+03) –	2.4982E+03 (1.6061E+03) –	1.7929E+03 (1.9112E+03) –	3.2653E+03 (1.7185E+03) –	**1.5950E+03 (1.6659E+03)** –
F23	5.2284E+02 (2.0595E+01) –	4.7956E+02 (1.1766E+01) –	4.5916E+02 (8.7508E+00) –	**4.3003E+02 (5.0786E+00)** =	4.3459E+02 (5.2143E+00) =	5.0490E+02 (1.5646E+01) –	4.3929E+02 (6.9001E+00) –
F24	5.8973E+02 (1.8663E+01) –	5.4197E+02 (7.6206E+00) –	5.3106E+02 (7.4577E+00) –	**5.0627E+02 (2.3317E+00)** =	5.0810E+02 (2.6001E+00) +	5.8374E+02 (1.6940E+01) –	5.1282E+02 (5.5948E+00) –
F25	5.7140E+02 (3.0541E+01) –	5.1923E+02 (3.4820E+01) –	5.0694E+02 (3.6446E+01) –	4.8530E+02 (1.6313E+01) =	4.8281E+02 (6.4445E+00) –	5.0912E+02 (3.1226E+01) –	**4.8034E+02 (1.0816E+00)** –
F26	2.5249E+03 (3.3797E+02) –	1.6146E+03 (1.2169E+02) –	1.4168E+03 (9.7281E+01) –	1.1443E+03 (4.4988E+01) +	**5.7211E+02 (4.0709E+02)** +	1.9319E+03 (2.8632E+02) –	1.2026E+03 (1.1870E+02) –
F27	7.1011E+02 (6.6568E+01) –	5.5080E+02 (2.3427E+01) –	5.4925E+02 (2.7842E+01) –	5.3361E+02 (1.9167E+01) –	5.3743E+02 (1.7376E+01) –	5.4355E+02 (1.7557E+01) –	**5.2543E+02 (9.2143E+00)** –
F28	4.9976E+02 (1.5368E+01) –	4.9185E+02 (2.0882E+01) –	4.7943E+02 (2.4173E+01) –	4.7322E+02 (2.2478E+01) –	4.7289E+02 (2.1643E+01) –	4.6481E+02 (1.5047E+01) –	**4.5913E+02 (1.1904E+01)** –
F29	5.1109E+02 (1.3726E+02) –	4.7761E+02 (8.0661E+01) –	4.8716E+02 (1.0502E+02) –	**3.5107E+02 (1.0430E+01)** =	3.6326E+02 (2.0650E+01) –	4.8938E+02 (1.1489E+02) –	**3.5289E+02 (9.7796E+00)** –
F30	8.0761E+05 (8.3375E+04) –	6.6809E+05 (9.2581E+04) –	6.8201E+05 (8.5144E+04) –	6.5396E+05 (7.3295E+04) +	6.5162E+05 (6.6381E+04) +	**5.8161E+05 (1.0211E+04)** +	6.5753E+05 (7.2413E+04) –
w/t/l	0/0/30	0/3/27	0/2/28	7/9/14	8/4/18	3/1/26	

and the configuration with larger probability is chosen. In the second half, the Cauchy distribution is used. Moreover, the covariance matrix learning with Euclidean neighborhood is used for adapting the values of the crossover operator. The proposed algorithm was tested on the benchmarks of the CEC2017 which is used in the Special Session and Competitions on single objective bound constrained real-parameter numerical Optimization of the IEEE CEC2017. When compared with other state-of-the-art DE algorithms taken from the literature, it shows a very competitive performance. For future work, more sensitivity analysis for algorithm parameters' will be conducted.

ACKNOWLEDGMENT

This paper was supported in partial by the deanship of research of Jordan University of Science and Technology, no. 517/2013.

REFERENCES

[1] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," J. Global Opt., vol. 11, no. 4, pp. 341–359, Dec. 1997.

[2] S.-M. Guo, J.S.-H. Tsai, C.-C. Yang, P.-H. Hsu, "A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set," In Proc. IEEE Congr Evol Comput, pp. 1003 - 1010, 2015.

[3] M.Z. Ali, N.H. Awad, P.N. Suganthan, "Multi-population Differential Evolution with Balanced Ensemble of Mutation Strategies for Large-Scale Global Optimization," Applied Soft Computing, V. 33, pp. 304-327, 2015.

[4] N.H. Awad, M.Z. Ali, R.G. Reynolds, "A differential evolution algorithm with success-based parameter adaptation for CEC2015 learning-based optimization," In Proc. IEEE Congr Evol Comput, pp. 1098 - 1105, 2015.

[5] M.Z. Ali, N.H. Awad, P.N. Suganthan, R.G. Reynolds, "An Adaptive Multipopulation Differential Evolution With Dynamic Population Reduction," IEEE Transactions on Cybernetics, Vol. PP, Issue. 99, PP.1-12, 2016.

[6] N.H Awad, M.Z Ali, P.N. Suganthan, E. Jaser, "A decremental stochastic fractal differential evolution for global numerical optimization," Information Sciences, Vol. 372, PP. 470-491, 2016.

[7] S. Das, S.S. Mullick, P.N. Suganthan, "Recent advances in differential evolution - An updated survey," Swarm and Evolutionary Computation, Vol. 27, pp. 1-30, 2016.

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

[9] N.H. Awad, M.Z. Ali, P.N. Suganthan, E. Jaser, "Differential Evolution with Stochastic Fractal Search Algorithm for Global Numerical Optimization," in Proc. IEEE Congr Evol Comput, 2016

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

[11] http://www.ntu.edu.sg/home/epnsugan/

[12] R. Tanabe, A. Fukunaga, "Improving the Search Performance of SHADE Using Linear Population Size Reduction," in IEEE CEC, 2014, pp. 1658 - 1665.

[13] R. Tanabe, A. Fukunaga, "Success-History Based Parameter Adaptation for Differential Evolution," in IEEE CEC, 2013, pp. 71–78.

[14] Y. Wang, H.-X. Li, T. Huang, L. Li, "Differential evolution based on covariance matrix learning and bimodal distribution parameter setting," Applied Soft Computing 18 (2014) 232–247.

[15] Y. Wang, Z. Cai, Q. Zhang, "Enhancing the search ability of differential evolution through orthogonal crossover," Information Sciences 185 (2012) 153–177.

[16] N.H. Awad, M.Z. Ali, P.N. Suganthan, R.G. Reynolds, "An Ensemble Sinusoidal Parameter Adaptation incorporated with L-SHADE for Solving CEC2014 Benchmark Problems," in Proc. IEEE Congr Evol Comput, 2016

[17] S.M. Elsayed, N.M. Hamza, R.A. Sarker, "Testing United Multi operator Evolutionary Algorithms-II on Single Objective Optimization Problems," in Proc. IEEE Congr. Evol. Comput., 2016

[18] N.H. Awad, M.Z. Ali, J.J. Liang, B.Y. Qu, P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization," Nanyang Technological University, Jordan University of Science and Technology and Zhengzhou University, Tech. Rep., 2016.

[19] I. Erlich, J.L. Rueda, S. Wildenhues, F. Shewarega, "Evaluating the Mean-Variance Mapping Optimization on the IEEE-CEC 2014 Test Suite," in Proc. IEEE Congr. Evol. Comput., 2014, pp. 1625 – 1632.

[20] J. Derrac, S. Garcia, D. Molina, F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,"Swarm Evol. Comput., vol. 1, no. 1, pp. 3–18, Mar. 2011.