# Modified L-SHADE for Single Objective Real-Parameter Optimization

Jia-Fong Yeh
Department of Computer Science and Information Engineering,
National Taiwan Normal University,
Taipei, Taiwan
60647005S@ntnu.edu.tw

Ting-Yu Chen
Department of Computer Science and Information Engineering,
National Taiwan Normal University,
Taipei, Taiwan
mazer0701@gmail.com

Tsung-Che Chiang
Department of Computer Science and Information Engineering,
National Taiwan Normal University,
Taipei, Taiwan
tcchiang@ieee.org

**<< This paper is included in the Proceedings of IEEE Congress on Evolutionary Computation (CEC 2019), held in New Zealand during Jun. 11-13, 2019. >>**

***Abstract*—In this paper we address single objective real parameter optimization by using differential evolution (DE). L-SHADE is a well-known DE with success history-based adaptation and linear population size reduction. We propose a modified L-SHADE (mL-SHADE), in which three modifications are made: (1) removal of the terminal value, (2) addition of polynomial mutation, and (3) proposal of a memory perturbation mechanism. Performance of the proposed mL-SHADE is verified by using ten benchmark functions in the CEC2019 100-Digit Challenge. The results show that mL-SHADE achieves a higher score than seven state-of-the-art adaptive evolutionary algorithms.**

***Keywords—differential evolution, adaptive, success history***

## I. Introduction

Many real-world problems can be formulated as real parameter optimization problems. Solving a $D$-dimensional real parameter optimization problem is to find the best decision vector $X = [x_1, x_2, \ldots, x_D]$ to maximize or minimize an objective function $f(X)$. Differential evolution (DE) is a kind of evolutionary algorithm (EA) and shows high potential in solving real parameter optimization problems in the literature. As an EA, DE searches for the optimal solution through repeating crossover, mutation, and selection. A DE has three parameters, $F$, $CR$, and $NP$, which denotes the scaling factor used in mutation, crossover rate, and the population size. Values of these parameters have great impact on the performance of DE and thus need to be determined carefully. Since manual tuning of parameter values is laborious and ineffective, developing DE with the ability of setting parameter values automatically and adaptively is an attractive and popular research topic.

This research is motivated by the 100-Digit Challenge Competition in CEC2019 [1]. The goal is to develop an algorithm to solve each of ten given problems to ten digits of accuracy without limit of computation time. In the last decade, many adaptive DE have been proposed for real-parameter optimization. In this study, we will develop a DE based on a recently proposed DE, called L-SHADE [2]. The rest of this paper is organized as follows. Section II reviews recent studies on adaptive EAs for solving single objective real parameter optimization, especially those developed based on L-SHADE. Section III elaborates the proposed mL-SHADE. Section IV presents experiments and results. Conclusions are made in Section V.

## II. Literature Review

The core idea in success-based adaptive DEs is to store successful $CR$ and $F$ values, i.e. the values by which the generated offspring (trial vector) can replace the parent (target vector), and then use these successful values to adjust the $CR$ and $F$ values/distributions to be used in later generations. JADE [3] and SaDE [4] are two representative algorithms. As the numeric study [5] pointed out, SaDE could not deal with the problems that require low and high $CR$ values simultaneously since it only stores a single mean $CR$ value.

Tanabe and Fukunaga proposed SHADE [6] based on JADE. The main feature of SHADE is to store multiple mean $CR$/$F$ values in the history memory. In each generation, the fitness-weighted Lehmer mean of successful $CR$/$F$ values will update the history memory in a round-robin way. To generate an offspring, SHADE first randomly selects a pair of $CR$ and $F$ values from the history memory, say $CR_r$ and $F_r$. Then, an actually used $CR$/$F$ value is generated by $randn(CR_r, 0.1)$/ $randc(F_r, 0.1)$, where $randn$ and $randc$ are normal distribution and Cauchy distribution, respectively. Finally, the offspring is generated by the current-to-$p$best/1/bin strategy with the actual $CR$/$F$ values. In the experiments, the SHADE outperformed several algorithms including JADE and CoDE [7].

Tanabe and Fukunaga later improved the SHADE by a linear population size reduction strategy (LPSR), proposing the L-SHADE [2]. The population size decreases linearly to four as the number of fitness function evaluation increases. The worst individuals in the population are removed when the population size decreases. In addition, the memory update mechanism is also modified.

Whenever a zero-value is used to update the *CR* memory, the memory cell is kept zero. Every time this memory cell is chosen, the actual *CR* value is zero, not a value taken from *randn*(0, 0.1). L-SHADE has very good performance and becomes the basis of many following algorithms.

Brest et al. [8] proposed iL-SHADE based on L-SHADE. JADE and its descendants all use the cur-to-*p*best mutation strategy, where one individual is randomly selected from the top *p*% of the current population as the *p*best. In iL-SHADE, the value of *p* begins with $p^{max}$ and linearly decreases to $p^{min}$ according to the number of fitness evaluations consumed. As for the memory update mechanism, iL-SHADE updates the memory value by the average of the old value and the incoming value. In addition, the last cell in the memory stores a pair of high *CR*/*F* values statically. When this cell is selected, the stored *CR*/*F* values are used directly without taking a value from the normal/Cauchy distributions. Values of other parameters were also adjusted, for example, the initial *CR* value was changed from 0.5 to 0.8 and the initial $r^{Ninit}$ value (which controls the initial population size) was changed from 18 to 12. iL-SHADE showed good performance and won the 3rd place in the CEC2016 competition. Brest et al. [9] continued improving iL-SHADE and proposed jSO. They set bound values of *CR* and *F* by deterministic rules based on the number of fitness function evaluations consumed. Values of some parameters such as initial *F* values in the history memory and the memory size were changed. In the CEC2018 competition, jSO won the second place.

The algorithm L-SHADE-EpSin, proposed by Awad et al. [10], is also based on L-SHADE. The control mechanism of *F* is divided into two stages. In the first stage, the value of *F* is generated by two sinusoidal functions; in the second stage, the control method is the same as that of L-SHADE. As L-SHADE-EpSin is based on L-SHADE, the population size decreases in the evolutionary process. When the population size reaches 20, ten individuals are generated randomly and then perform Gaussian walks for 250 iterations. These individuals will replace the worst individuals in the population. In the CEC2016 competition, L-SHADE-EpSin won the second place. Awad et al. [11] kept improving L-SHADE-EpSin and proposed L-SHADE-cnEpSin. Instead of randomly selecting between the two sinusoidal functions, they introduced the learning strategy from SaDE to select the sinusoidal functions. Besides, covariance matrix learning was applied in a certain probability. The local search procedure in L-SHADE-EpSin was removed. This new algorithm won the third place in the CEC2017 competition.

Stanovov et al. [12] proposed L-SHADE-RSP, which is also an extension of L-SHADE. They introduced the rank selection into the cur-to-*p*best strategy. Specifically, the top *p*% individuals are selected as the pbest in the probability proportional to their ranks, not just randomly as in L-SHADE. They also adopted the bounding rule of jSO to control *CR*. L-SHADE-RSP won the 2nd place in the CEC2018 competition. The experimental results showed its potential in solving higher dimensional problems.

Kumar et al. [13] proposed EBOwithCMAR, which combines effective butterfly optimizer (EBO) [14] and a covariance matrix adapted retreat (CMAR) phase. This algorithm divides the population into sub-populations. It also adopts the mechanisms of L-SHADE and sequential quadratic programming (SEQ). Values of *F* are generated by tangential approaches. It won the first place in the CEC2017 competition.

Zhang and Shi [15] proposed the hybrid sampling evolution strategy (HS-ES). It integrates CMA-ES, a multivariate sampling method, and UMDAc, a univariate sampling method, by a cascade model. UMDAc is executed first, and then CMA-EA is executed. Based on the results, values of some variables are fixed. At last, UMDAc is executed again. HS-ES combines the advantages of multivariate and univariate methods and won the first place in the CEC2018 competition.

## III. Proposed mL-SHADE Algorithm

### A. Overview

In this paper we propose a modified L-SHADE (mL-SHADE) algorithm. Table I presents the pseudo code of mL-SHADE. Three modifications are briefly described here:

*1) Removal of the "terminal" value:* In L-SHADE the terminal value $\perp$ is put in the *CR* memory when the incoming *CR* value is zero. Thereafter, the terminal value cannot be modified, and *CR* is always set by zero when the terminal value is chosen. We found that this mechanism is not good for solving some functions in the CEC2019 competition (e.g. $f_4$) and hence removed it. The removal gives the SHA mechanism more chance to adjust *CR* to suitable values.

*2) Addition of polynomial mutation:* We apply the polynomial mutation [16] to the trial vector in probability $m_r$ to introduce randomness and increase diversity. (Check lines 12-17 in Table I.)

*3) Proposal of a memory perturbation mechanism:* Since we observed that the history memory may keep unchanged for a long time and cause the search process to get stuck, we do perturbation to the memory after $N^{stuck}$ generations without update. (Check lines 38-42 in Table I.)

### B. Initialization

The initial population are generated by uniform random initialization within the range of variables, as the rule of CEC2019 competition requires.

TABLE I. PSEUDO CODE OF THE PROPOSED ML-SHADE

Notations:

$D$: problem dimension
$N^{init}$: size of the initial population
$A$: the archive of inferior solutions
$N^{stuck}$: the maximum number of generations allowed no memory update
$M_{CR}$, $M_F$: the history memory of mean $CR$, $F$ values
$H$: size of the history memory
$S_{CR}$, $S_F$, $\Delta_f$: the archive of successful $CR$, $F$, and fitness improvement
*randn*: normal distribution
*randc*: Cauchy distribution
$m_r$: mutation rate for polynomial mutation

$g = 1, N_g = N^{init}, \boldsymbol{A} = \varnothing$;
$m_r = 0.05, stuck = 0$; $k = 1$
Initialize population $\boldsymbol{P} = (x_{1,g}, \cdots, x_{N,g})$ randomly;
Set all values in $M_{CR}$, $M_F$ to 0.5;
**while** *the terminating criteria are not met* **do**
  $S_{CR} = \emptyset, S_F = \emptyset$ ;
  **for** $i = 1$ **to** $N_g$ **do**
    $r_i$ = Select from $[1, H]$ randomly;
    $CR_{i,g} = \text{randn}_i(M_{CR,r_i}, 0.1)$;
    $F_{i,g} = \text{randc}_i(M_{F,r_i}, 0.1)$;
    Generate trial vector $u_{i,g}$ by current-to-*p*best/1/bin;
    **if** rand[0,1) $\leq m_r$ **then**
      $pm\_u_{i,g} = \text{PolynomialMutation}(u_{i,g}, 1.0/D, 10.0)$;
      **if** $f(pm\_u_{i,g}) \leq f(u_{i,g})$ **then**
        $u_{i,g} = pm\_u_{i,g}$;
      **end if**
    **end if**
  **end for**
  **for** $i = 1$ **to** $N_g$ **do**
    **if** $f(u_{i,g}) \leq f(x_{i,g})$ **then**
      $x_{i,g+1} = u_{i,g}$;
    **else**
      $x_{i,g+1} = x_{i,g}$;
    **end if**
    **if** $f(u_{i,g}) < f(x_{i,g})$ **then**
      $\boldsymbol{A} = \boldsymbol{A} \cup \{x_{i,g}\}$;
      $S_{CR} = S_{CR} \cup \{CR_{i,g}\}, S_F = S_F \cup \{F_{i,g}\}$;
      $\Delta_f = \Delta_f \cup \{|f(u_{i,G}) - f(x_{i,G})|\}$
    **end if**
  **end for**
  **if** $|\boldsymbol{A}| > |\boldsymbol{P}|$ **then**
    Resize $\boldsymbol{A}$ by removing $|\boldsymbol{A}| - |\boldsymbol{P}|$ individuals randomly;
  **end if**
  **if** $S_{CR} \neq \emptyset$ **and** $S_F \neq \emptyset$ **then**
    Update the $k^{\text{th}}$ memory in $M_{CR}$ and $M_F$ by Eq. (4)-(6);
    $k = k$ mod $H + 1$;
  **else**
    $stuck = stuck + 1$;
    **if** $stuck \geq N^{stuck}$ **then**
      Perturb the $k^{\text{th}}$ memory in $M_{CR}$ and $M_F$ by Eq. (7)-(8);
      $stuck = 0$; $k = k$ mod $H + 1$;
    **end if**
  **end if**
  Calculate $N_{g+1}$ according to LPSR;
  **if** $N_{g+1} < |\boldsymbol{P}|$ **then**
    Resize $\boldsymbol{P}$ by removing $|\boldsymbol{P}| - N_{g+1}$ worst individuals;
    Resize $\boldsymbol{A}$ by removing $|\boldsymbol{A}| - N_{g+1}$ individuals randomly;
  **end if**
  $g = g + 1$;
**end while**

### C. Mutation Strategy

We use the current-to-*p*best/1 mutation to generate the mutant vector $v_{i,g}$, as defined in (1). Every individual in the population serves as the target vector $x_{i,g}$ once. The individual $x_{pbest,g}$ is randomly selected from the top $p$% (in terms of the fitness value) of the population; the individual $x_{r1,g}$ is selected randomly from the entire population; the individual $x_{r2,g}$ is selected randomly from the union of the population and the archive of inferior solutions. (When a target vector is replaced by a trial vector, the target vector is added into the archive.) We ensure that $x_{r1,g}$ and $x_{r2,g}$ are different individuals.

$$v_{i,g} = x_{i,g} + F_i \cdot \left(x_{pbest,g} - x_{i,g}\right) + F_i \cdot \left(x_{r1,g} - x_{r2,g}\right) \quad (1)$$

When the mutation strategy generates values $v_{j,i,g}$ of variables out of the bounds $[x_j^{min}, x_j^{max}]$, we repair the values by the method of JADE, as presented in (2).

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

### D. Crossover

We use the binomial crossover to generate the trial vector $u_{i,g}$, as defined in (3). The function *rand*[0, 1) returns a random value between 0 and 1 (excluding 1). $j_{rand}$ is a random value in [1, $D$], where $D$ is the problem dimension.

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & if\ rand[0,1) \le CR_i \text{ or } j = j_{rand} \\ x_{j,i,g} & otherwise \end{cases} \quad (3)$$

### E. Selection

After generating a trial vector $u_{i,G}$ from a target vector $x_{i,G}$ by the current-to-*p*best/1 mutation and the binomial crossover, $x_{i,G}$ will be replaced by $u_{i,G}$ if $u_{i,G}$ is not worse than $x_{i,G}$. Please check lines 20-24 in Table I.

### F. Parameter Control

Our algorithm is based on L-SHADE, and we use its success history-based adaption and LPSR strategies. The history memory $M_{CR}$ and $M_F$ stores potential mean values of $CR$ and $F$. In Table I, lines 8-11 presents how values of $CR_{i,g}$ and $F_{i,g}$ are obtained based on the memory to generate new solutions $u_{i,g}$. We also repair their values by the same method of L-SHADE. When the value of $CR_{i,g}$ is out of the bound [0, 1], its value is fixed to the closer boundary value. When the value of $F_{i,g}$ is not greater than zero, we take another value by the Cauchy distribution again.

We update the history memory in the same way as L-SHADE does, as (4)-(6) show. Every time a trial vector produced by a pair of $CR$/$F$ values replaces the target vector, we store the $CR$ and $F$ values in $S_{CR}$ and $S_F$, respectively. We also store the improvement $\Delta f_k$, where $k$ means that it is the $k^{\text{th}}$ successful $CR$/$F$ values. At the end of each generation, we calculate the weight of each successful $CR$/$F$ value by (5). Finally, we update the history memory by the weighted Lehmer mean, defined in (6). In (5) and (6), $S$ can denote $S_{CR}$ or $S_F$, and $S_k$ denotes the $k^{\text{th}}$ value in $S$.

$$\Delta f_k = \left|f\left(u_{i,G}\right) - f\left(x_{i,G}\right)\right| \quad (4)$$

$$w_k = \frac{\Delta f_k}{\sum_{l=1}^{|S|} \Delta f_l} \quad (5)$$

$$mean_{wL}(S) = \frac{\sum_{k=1}^{|S|} w_k \cdot S_k^2}{\sum_{k=1}^{|S|} w_k \cdot S_k} \quad (6)$$

Recall that we remove the terminal value in L-SHADE. First, when the value of the selected memory $M_{CR,ri}$ is zero, we still generate $CR_{i,g}$ by *randn*(0, 0.1). Second, the $CR$ memory with a zero value can still be updated by incoming new values.

As mentioned in Section III-A, we count the number of consecutive generations without updating the history memory. When the number reaches a predefined threshold $N^{stuck}$, we perturb the memory by (7) and (8). The idea is to change the value to the opposite end. If the original value is small, we change it to a large value.

$$M_{CR,k} = 1.0 - M_{CR,k} \quad (7)$$

$$M_{F,k} = 1.0 - M_{F,k} \quad (8)$$

### G. Terminating Criteria

Our algorithm stops when one of the following two criteria is satisfied: (1) when the error to the optimum (known for each function in the CEC2019 100-Digit Challenge) is smaller than $10^{-9}$, i.e., we get the 10-point score (the perfect score for one test function) in the challenge; (2) when the maximum number of fitness function evaluation is reached.

## IV. Experiments and Results

### A. Benchmark Functions

There are ten test functions in the CEC2019 100-Digit Challenge. The optimum value of every tested functions is one. Table II summarizes the dimensionality and search ranges of these ten functions. Their definitions are included in Appendix.

TABLE II. Scale of CEC2019 100-Digit Challenge Benchmark Functions

| No. | Function Name | $D$ | Search Range |
|---|---|---|---|
| 1 | Storn's Chebyshev Polynomial Fitting Problem | 9 | [-8192, 8192] |
| 2 | Inverse Hilbert Matrix Problem | 16 | [-16384, 16384] |
| 3 | Lennard-Jones Minimum Energy Cluster | 18 | [-4, 4] |
| 4 | Rastrigin's Function | 10 | [-100, 100] |
| 5 | Griewangk's Function | 10 | [-100, 100] |
| 6 | Weierstrass Function | 10 | [-100, 100] |
| 7 | Modified Schwefel's Function | 10 | [-100, 100] |
| 8 | Expanded Schaffer's F6 Function | 10 | [-100, 100] |
| 9 | Happy Cat Function | 10 | [-100, 100] |
| 10 | Ackley Function | 10 | [-100, 100] |

### B. Parameter Setting

Since our mL-SHADE is based on L-SHADE, we need to set values for all parameters of L-SHADE. Table III lists our setting. Values of $N^{init}$, $N_{min}$, $H$, $p$, and initial $M_{CR}/M_F$ memory were set by the same values in L-SHADE [2]. We call the polynomial mutation probabilistically and need to set three parameters, $m_r$, $p_m$, and $\eta$. We did preliminary experiments on $r^{arc}$, $m_r$, and $\eta$ to determine their values. We need one more parameter $N^{stuck}$ in our history memory perturbation mechanism. According to the rule of the CEC2019 100-Digit Challenge, we can have at most two parameters with function-dependent values. Since the value of $N^{stuck}$ is influential on the performance, we set its values by functions. Table IV lists its values. The 100-Digit Challenge does not define the limit of computation time or number of fitness function evaluations. We observed the convergence curves of our mL-SHADE and set the maximum number of fitness evaluations by $2 \cdot 10^6$.

TABLE III. Values of Parameters in the Proposed mL-SHADE

| Parameter | Meaning | Value |
|---|---|---|
| $N^{init}$ | size of the initial population | $18 \cdot D$ |
| $N_{min}$ | minimal population size | 4 |
| $H$ | size of the history memory | 6 |
| $r^{arc}$ | archive size $\lvert A \rvert$ = round($r^{arc} \cdot N^{init}$) | 1.0 |
| $p$ | required in the cur-to-$p$best/1 mutation | 0.11 |
| $M_{CR}{}^0/M_F{}^0$ | initial values of $CR/F$ memory | 0.5 |
| $m_r$ | probability of polynomial mutation | 0.05 |
| $p_m$, $\eta$ | parameters of polynomial mutation | $1/D$, 10 |
| $MaxNFE$ | maximum number of fitness evaluations | $2 \cdot 10^6$ |

TABLE IV. Function-Specific Parameter Values

| No. | $N^{stuck}$ | No. | $N^{stuck}$ |
|---|---|---|---|
| 1 | 400 | 6 | 400 |
| 2 | 400 | 7 | 400 |
| 3 | 6 (same as $H$) | 8 | 400 |
| 4 | 400 | 9 | 6 (same as $H$) |
| 5 | 400 | 10 | 400 |

### C. Benchmark Algorithms

We compared our mL-SHADE with seven state-of-the-art EAs: HS-ES, LSHADE-RSP, ELSHADE-SPACMA, EBOwithCMAR, jSO, LSHADE-cnEpSin, and L-SHADE. The first six algorithms were the best performers in the previous CEC2018 and CEC2017 real-parameter competitions. We downloaded the source codes of these algorithms from the organizer's website [17]. We re-implemented L-SHADE. Since there is some difference between the rules of previous competitions and of CEC2019 100-Digit Challenge, we did some small modifications to the codes. First, we removed their stopping criterion that stops when the error to the optimum is smaller than $10^{-8}$. Second, we did some fix in EBOwithCMAR and LSHADE-cnEpSin for the population size: (1) the

archive size is rounded to the nearest integer, for example, it is $1.4\times18\times9 = 226.8 \approx 227$ when solving $f_1$; (2) the code of EBOwithCMAR cannot generate enough *CR* values (*D* values are required but only $D-1$ are generated) when the problem dimension *D* is an odd number. In this case, we duplicate the last CR value. Third, we fixed the value of parameter $G_{max}$ in LSHADE-cnEpSin by 2163, which was used by LSHADE-cnEpSin to solve 10-*D* problems.

### *D. Performance Metric*

We evaluated the eight algorithms by the scoring criterion of the CEC2019 100-Digit Challenge [1]. Each algorithm solved each function by 50 runs. The total number of correct digits $N_c$ in the 25 runs with the lowest function values is counted. The score for a function is then $N_c/25$. The perfect score for one function is 10, and the perfect score for the challenge is 100, which is achieved when the best 25 out of 50 runs for all ten functions give the minimum to 10-digit accuracy.

The minimum for all functions to ten digits of accuracy is 1.000000000. When an algorithm finds solutions with objective values 2.000000000, 1.924235666, and 1.003243567, it scores 0, 1, and 3 points, respectively.

### *E. Performance Analysis*

In Table V, we summarize the number of runs with respect to the number of corrected digits after our mL-SHADE solved ten test functions for 50 runs. mL-SHADE solved functions $f_1$, $f_2$, $f_3$, and $f_6$ perfectly in all 50 runs. For functions $f_4$, $f_5$, and $f_{10}$, mL-SHADE sometimes got stuck but still succeeded in at least 43 of 50 runs. Functions $f_7$, $f_8$, and $f_9$ are difficult to mL-SHADE.

TABLE V. FIFTY RUNS FOR EACH FUNCTION SORTED BY THE NUMBER OF CORRECT DIGITS

| No. | Number of correct digits | | | | | | | | | | | Score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 50 | 10 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 50 | 10 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 50 | 10 |
| 4 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 48 | 10 |
| 5 | 0 | 0 | 2 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 44 | 10 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 50 | 10 |
| 7 | 0 | 21 | 19 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 9 | 5.04 |
| 8 | 1 | 49 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 9 | 0 | 0 | 46 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2.16 |
| 10 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 43 | 10 |
| | | | | | | | | | | | Total: | 78.2 |

In Table VI, we present the scores of all eight algorithms for all ten functions. The highest score of each function is marked in bold. Our mL-SHADE and EBOwithCMAR are the best two algorithms, both got the highest scores for eight functions. In the experiments, we found that L-SHADE-based algorithms did not perform well in solving $f_4$, which could be caused by the terminal value in L-SHADE. By removing the terminal value, our mL-SHADE solved $f_4$ successfully. EBOwithCMAR also solved $f_4$ well since it is not based on L-SHADE. For the three difficult functions $f_7$, $f_8$, and $f_9$, no single algorithm can get the highest score for more than one of them. It is worth more investigations on the relationship between algorithm design and problem characteristics.

## V. CONCLUSIONS

In this paper we proposed an adaptive DE to solve real-parameter optimization problems. We examined the search behaviors of a recent algorithm, L-SHADE, and then developed a modified version by fixing the observed weakness. We removed the terminal value to prevent the parameter control mechanism from sticking with zero *CR* values. We added the polynomial mutation to increase diversity in the decision space. We also proposed a memory perturbation mechanism to increase diversity in the parameter space. We tested the proposed mL-SHADE on the ten functions in CEC2019 100-Digit Challenge and compared its performance with seven state-of-the-art algorithms. Our algorithm got the highest scores in eight out of ten functions and also the highest total score.

In our experiments, we found three functions that are difficult to all tested algorithms. No single algorithm can get the highest score for more than one of them. We will continue our research to study how different algorithms fit different functions and then to propose a better integration. Other research directions include developing an adaptive method to adjust the value of $N^{stuck}$ in our memory perturbation mechanism and a new method to update the *CR*/*F* values in the memory. Last, the mL-SHADE have many hyper-parameters. We should examine whether performance of mL-SHADE is sensitive to the values of them.

TABLE VI. SCORES OF TESTED ALGORITHMS

| No. | mL-SHADE | L-SHADE | ELSHADESPACMA | jSO |
|---|---|---|---|---|
| 1 | **10** | **10** | **10** | **10** |
| 2 | **10** | **10** | **10** | **10** |
| 3 | **10** | 7.16 | 5.44 | **10** |
| 4 | **10** | 0.24 | 0.84 | 3.88 |
| 5 | **10** | **10** | **10** | **10** |
| 6 | **10** | **10** | **10** | **10** |
| 7 | **5.04** | 1 | 1.08 | 1 |
| 8 | 1 | 1 | 1.16 | 1.08 |
| 9 | 2.16 | 2.04 | **3** | 2.12 |
| 10 | **10** | **10** | 7.88 | **10** |
| Score | **78.2** | 61.44 | 59.4 | 68.08 |
| No. | LSHADE -RSP | LSHADE -cnEpSin | EBOwith CMAR | HS-ES |
| 1 | **10** | **10** | **10** | 7.6 |
| 2 | **10** | **10** | **10** | 0 |
| 3 | **10** | 6.12 | **10** | 1.72 |
| 4 | 6.76 | 4.6 | **10** | 4.96 |
| 5 | **10** | **10** | **10** | **10** |
| 6 | **10** | **10** | **10** | **10** |
| 7 | 1.36 | 0.84 | 0.56 | 0.16 |
| 8 | 1 | 1.04 | **1.68** | 0.16 |
| 9 | 2.2 | 2.36 | 2.08 | **3** |
| 10 | **10** | **10** | **10** | **10** |
| Score | 71.32 | 64.96 | 74.32 | 47.6 |

## REFERENCES


[1] K. V. Price, N. H. Awad, M. Z. Ali, P. N. Suganthan, "Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization," Technical Report, Nanyang Technological University, Singapore, November 2018.

[2] R. Tanabe and A. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," In: Proceedings of IEEE Congress on Evolutionary Computation (CEC2014), pp. 1658−1665, 2014. [L-SHADE]

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

[4] 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. [SaDE]

[5] C. A. Chen and T. C. Chiang, "Adaptive differential evolution: a visual comparison," In: Proceedings of IEEE Congress on Evolutionary Computation (CEC2015), pp. 401−408, 2015.

[6] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," In: Proceedings of IEEE Congress on Evolutionary Computation (CEC2013), pp. 71-78, 2014. [SHADE]

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

[8] J. Brest, M. S. Maučec, and B. Bošković, "iL-SHADE: improved LSHADE algorithm for single objective real-parameter optimization," In: Proceedings of IEEE Congress on Evolutionary Computation (CEC2016), pp. 1188–1195, 2016. [iL-SHADE]

[9] J. Brest, M. S. Maučec, and B. Bošković, "Single objective real-parameter optimization: algorithm jSO," In: Proceedings of IEEE Congress on Evolutionary Computation (CEC2017), pp. 1311–1318, 2017. [jSO]

[10] 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: Proceedings of IEEE Congress on Evolutionary Computation (CEC2016), pp. 2958-2965, 2016. [L-SHADE-EpSin]

[11] N.H. Awad, M.Z. Ali, P.N. Suganthan, R.G. Reynolds, "Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems," In: Proceedings of IEEE Congress on Evolutionary Computation (CEC2017), pp. 372-379, 2017. [L-SHADE-cnEpSin]

[12] V. Stanovov, S. Akhmedova and E. Semenkin, "LSHADE algorithm with rank-based selective pressure strategy for solving CEC 2017 benchmark problems," In: Proceedings of IEEE Congress on Evolutionary Computation (CEC2018), pp. 1-8, 2018. [L-SHADE-RSP]

[13] A. Kumar, R. K. Misra and D. Singh, "Improving the local search capability of effective butterfly optimizer using covariance matrix adapted retreat phase," In: Proceedings of IEEE Congress on Evolutionary Computation (CEC2017), pp. 1835-1842, 2017. [EBOwithCMAR]

[14] A. Kumar, R. K. Misra, and D. Singh, "Butterfly optimizer," In: Proceedings of IEEE Workshop on Computational Intelligence: Theories, Applications, and Future Directions, pp. 1-6, 2015.

[15] G. Zhang and Y. Shi, "Hybrid sampling evolution strategy for solving single objective bound constrained problems," In: Proceedings of IEEE Congress on Evolutionary Computation (CEC2018) , pp. 1-7, 2018. [HS-ES]

[16] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," Computer Science and Informatics, 1996.

[17] Benchmarks for Evaluation of Evolutionary Algorithms, http://www.ntu.edu.sg/home/epnsugan/index_files/cec-benchmarking.htm, accessed on 2019/1/20.

## APPENDIX

TABLE VII. DEFINITIONS OF CEC2019 100-DIGIT CHALLENGE BENCHMARK FUNCTIONS

| | |
|---|---|
| Storn's Chebyshev Polynomial Fitting Problem | $f_1(\mathrm{x}) = p_1 + p_2 + p_3,$<br>$p_1 = \begin{cases}(u-d)^2 \; if \; u < d, \\ 0 \qquad otherwise;\end{cases} \quad u = \sum_{j=1}^{D} x_j (1.2)^{D-j}$<br>$p_2 = \begin{cases}(v-d)^2 \; if \; v < d, \\ 0 \qquad otherwise;\end{cases} \quad v = \sum_{j=1}^{D} x_j (-1.2)^{D-j}$<br>$p_k = \begin{cases}(w_k-1)^2 \; if w_k > 1 \\ (w_k+1)^2 \; if w_k < 1 \\ 0 \qquad otherwise;\end{cases} \quad w_k = \sum_{j=1}^{D} x_j \left(\frac{2k}{m} - 1\right)^{D-j}$<br>$p_3 = \sum_{k=0}^{m} p_k, \qquad k = 0,1,\dots,m, \qquad m = 32D.$<br>$d = 72.661$ for $D = 9$ |
| Inverse Hilbert Matrix Problem | $f_2(\mathrm{x}) = \sum_{i=1}^{n}\sum_{k=1}^{n} \lvert w_{i,k} \rvert$<br>$(w_{i,k}) = \mathbf{W} = \mathbf{HZ} - \mathbf{I}, \qquad \mathbf{I} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$<br>$\mathbf{H} = (h_{i,k}), h_{i,k} = \frac{1}{i+k-1}, i,k = 1,2,\dots,n, n = \sqrt{D}$<br>$\mathbf{Z} = (z_{i,k}), \qquad z_{i,k} = x_{i+n(k-1)}$ |
| Lennard-Jones Minimum Energy Cluster | $f_3(\mathrm{x}) = 12.7120622568 + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \left(\frac{1}{d_{i,j}^2} - \frac{2}{d_{i,j}}\right),$<br>$d_{i,j} = (\sum_{k=0}^{2} (x_{3i+k-2} - x_{3j+k-2})^2)^3, n = D/3$ |
| Rastrigin's Function | $f_4(\mathrm{x}) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$ |
| Griewangk's Function | $f_5(\mathrm{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ |
| Weierstrass Function | $f_6(x) = \sum_{i=1}^{D} \left( \sum_{k=0}^{k_{max}} \left[a^k \cos\left(2\pi b^k (x_i + 0.5)\right)\right] \right) - D \sum_{k=0}^{k_{max}} a^k \cos(\pi b^k)$<br>$a = 0.5, \;\; b = 3, \;\; k_{max} = 20$ |
| Modified Schwefel's Function | $f_7(x) = 418.9829D - \sum_{i=1}^{D} g(z_i)$<br>$z_i = x_i + 420.9687462275036$<br>$g(z_i) = \begin{cases} z_i \sin\left(\lvert z_i \rvert^{1/2}\right) & if \; \lvert z_i \rvert \le 500 \\ \left(500 - mod(z_i, 500)\right) \sin\left(\sqrt{\lvert 500 - mod(z_i, 500) \rvert}\right) - \frac{(z_i - 500)^2}{10000D} & if \; z_i > 500 \\ \left(mod(\lvert z_i \rvert, 500) - 500\right) \sin\left(\sqrt{\lvert mod(z_i, 500) - 500 \rvert}\right) - \frac{(z_i + 500)^2}{10000D} & if \; z_i < -500 \end{cases}$ |
| Expanded Schaffer's F6 Function | $f_8(x) = g(x_1, x_2) + g(x_2, x_3) \dots + g(x_{D-1}, x_D) + g(x_D, x_1)$<br>$g(x,y) = 0.5 + \frac{\sin^2\left(\sqrt{x^2+y^2}\right) - 0.5}{\left(1 + 0.001(x^2+y^2)\right)^2}$ |
| Happy Cat Function | $f_9(x) = \left\lvert \sum_{i=1}^{D} x_i^2 - D \right\rvert^{1/4} + \left(0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i\right) \Big/ D + 0.5$ |
| Ackley Function | $f_{10}(x) = -20exp\left(0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e$ |