2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Improving the Search Performance of SHADE Using Linear Population Size Reduction

Ryoji Tanabe and Alex S. Fukunaga
Graduate School of Arts and Sciences
The University of Tokyo

***Abstract*—SHADE is an adaptive DE which incorporates success-history based parameter adaptation and one of the state-of-the-art DE algorithms. This paper proposes L-SHADE, which further extends SHADE with Linear Population Size Reduction (LPSR), which continually decreases the population size according to a linear function. We evaluated the performance of L-SHADE on CEC2014 benchmarks and compared its search performance with state-of-the-art DE algorithms, as well as the state-of-the-art restart CMA-ES variants. The experimental results show that L-SHADE is quite competitive with state-of-the-art evolutionary algorithms.**

I. Introduction

Differential Evolution (DE) is an Evolutionary Algorithm (EA) that was primarily designed for real parameter optimization problems [1]. Despite its relative simplicity, DE has been shown to be competitive with more complex optimization algorithms, and has been applied to many practical problems [2]. However, the search performance of DE algorithms depends on control parameter settings [2]. Where a standard DE has three main control parameters, which are the population size N, scaling factor F, and crossover rate CR. Unfortunately, it is well-known that the optimal settings of these parameters are problem-dependent. Therefore, when applying DE to a real-world problem, it is often necessary to tune the control parameters in order to obtain the desired results. Since this is a significant problem in practice, adaptive mechanisms for adjusting the control parameters on-line during the search process have been studied by many researchers [3]–[6].

Success-history based adaptation is a novel mechanism for parameter adaptation based on a historical memory of successful parameter settings that were previously used found during the run [7]–[9]. Success-history based adaptation uses a *historical memory* M_{CR}, M_F which stores a set of CR, F values that have performed well in the past, and generate new CR, F pairs by directly sampling the parameter space close to one of these stored pairs. Success-History based Adaptive DE (SHADE) [7]–[9] is an improved version of JADE [5] which uses a different parameter adaptation mechanism based on this success-history based adaptation. This paper uses SHADE 1.1 [9], the latest version of the SHADE algorithm. SHADE 1.1 dominates the SHADE 1.0 algorithm proposed in two earlier conference papers [7], [8].

While SHADE 1.0 and SHADE 1.1 automatically adjust the CR and F parameters, the population size N remains constant throughout the search. The size of the population used by EA plays a significant role in controlling the rate of convergence. Small population sizes tend to result in faster convergence, but increases the risk of converging to a local optimum. On the other hand, large population sizes can encourage wider exploration of the search space, but the rate of convergence tends to be slower. The optimal population size depends on complex interaction of a number of factors, including the problem to which the EA is being applied, as well as the values of other control parameters such as the mutation rate. Thus, adaptive population resizing methods have been an active area of research (see [10] for a survey).

However, as noted in [10], adaptive population resizing methods such as GAVaPS [11] tend to replace one control parameter (population size) with multiple, meta-level control parameters, and are difficult to use in practice due to the need to tune the meta-level control parameters. Unlike adaptive methods for other control parameters such as mutation rate and crossover rate, successful adaptation of population size has proven difficult. Thus, in recent years, population resizing methods based on simple, deterministic rules (as opposed to adaptive schemes) have been proposed. These approaches have been found to be highly effective for improving EA performance. Representative examples include IPOP-CMA-ES [12], GL-25 [13], IPSO [14], Dynamic Population Size Reduction (DPSR) [15], and Simple Variable Population Sizing (SVPS) [16]. In contrast to adaptive methods which can both increase or decrease the population size depending on the state of the search, these deterministic methods either monotonically increase or decrease the population size, based on predetermined conditions. This significantly simplifies the behavior of the population resizing methods.

To further enhance the performance of SHADE 1.1, we propose L-SHADE, which incorporates Linear Population Size Reduction (LPSR), a simple deterministic population resizing method which continuously reduces the population size in accordance with a linear function. Our LPSR method is a simplified, special case of SVPS [16] which reduces the population linearly, and requires only 1 parameter (initial population sizes). DPSR [15] reduces the population by half at predetermined intervals and has 2 parameters: the initial population size and the frequency of the population reduction r_{freq} which has to be tuned to match the initial population size as well as the dimensionality of the problem. SVPS is a more general framework in which the shape of the population size reduction schedule is determined according to control parameters τ and ρ. In addition to τ and ρ, which influence the shape of the population size curve, SVPS requires the initial and final population size at the end of the run. In contrast to DPSR, which drastically reduces (halves) the population at discrete, intervals, SVPS continuously reduces the population size. Compared with both methods, LPSR is very simple and

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

Index	1	2	...	$H-1$	H
M_{CR}	$M_{CR,1}$	$M_{CR,2}$	...	$M_{CR,H-1}$	$M_{CR,H}$
M_F	$M_{F,1}$	$M_{F,2}$	...	$M_{F,H-1}$	$M_{F,H}$

Fig. 1: The historical memory M_{CR}, M_F

has the advantage in that there are fewer control parameters which the user must tune.

This paper evaluates our proposed method, called L-SHADE, on the 30 benchmark functions from the *CEC2014 Special Session on Real-Parameter Single Objective Optimization benchmark suite* [17]. We show experimentally that L-SHADE significantly improves upon the performance of SHADE 1.1 and also outperforms previous DE variants, including JADE [5], CoDE [18], EPSDE [6], SaDE [4], and dynNP-jDE [15]. In addition, we compared L-SHADE with NBIPOP-$_{\mathrm{A}}$CMA-ES [19] and iCMAES-ILS [20] which are state-of-the-art restart CMA-ES [12] variants and co-winners of competition on real-parameter single objective Optimization at CEC 2013. Our results show that L-SHADE is highly competitive with these CMA-ES variants.

II. Success-History based Adaptive DE with Linear Population Size Reduction

This section describes L-SHADE. We first describe SHADE 1.1 [9] in Sections II-A to II-E. Then, in Section II-F, we describe LPSR and propose L-SHADE, which is SHADE 1.1 extended with LPSR.

Similar to other evolutionary algorithms for numerical optimization, a DE population is represented as a set of real parameter vectors $\boldsymbol{x}_i = (x_1, ..., x_D)$, $i = 1, ..., N$, where D is the dimensionality of the target problem, and N is the population size. At the beginning of the search, the individual vectors $\boldsymbol{x}_i$ in population are initialized randomly. Then, a process of trial vector generation and selection are repeated until some termination criterion is encountered.

A. Control parameters assignments based on historical memory

As shown in Figure 1, SHADE 1.1 maintains a historical memory with H entries for both of the DE control parameters CR and F, M_{CR}, M_F. The scaling factor $F \in [0, 1]$ controls the magnitude of the differential mutation operator and $CR \in [0, 1]$ is the crossover rate. In the beginning, the contents of $M_{CR,k}, M_{F,k}$ $(k = 1, ..., H)$ are all initialized to 0.5. In each generation G, the control parameters CR_i and F_i used by each individual $\boldsymbol{x}_i$ are generated by randomly selecting an index r_i from $[1, H]$, and then applying the formulas below:

$$CR_i = \begin{cases} 0 & \text{if } M_{CR,r_i} = \bot. \\ \mathrm{randn}_i(M_{CR,r_i}, 0.1) & \text{otherwise} \end{cases} \quad (1)$$

$$F_i = \mathrm{randc}_i(M_{F,r_i}, 0.1) \quad (2)$$

In case a value for CR_i outside of $[0, 1]$ is generated, it is replaced by the limit value (0 or 1) closest to the generated value. When $F_i > 1$, F_i is truncated to 1, and when $F_i \leq 0$, Eq. (2) is repeatedly applied to try to generate a valid value. These manners are according to the procedure for JADE [5]. In Eq. (1), if M_{CR,r_i} has been assigned the "terminal value" $\bot$, CR_i is set to 0.

B. Reproduction of trial vectors by using current-to-pbest/1/bin

After the control parameter values CR_i and F_i are assigned for each individual $\boldsymbol{x}_{i,G}$, a mutant vector $\boldsymbol{v}_{i,G}$ is generated from an existing population members by applying the current-to-$p$best/1 mutation strategy, which is the mutation strategy used by JADE [5], and a variant of the traditional current-to-best/1 strategy where the greediness is adjustable using a parameter p:

$$\begin{aligned} \boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} &+ F_i \cdot (\boldsymbol{x}_{pbest,G} - \boldsymbol{x}_{i,G}) \\ &+ F_i \cdot (\boldsymbol{x}_{r1,G} - \boldsymbol{x}_{r2,G}) \end{aligned} \quad (3)$$

In Eq. (3), individual $\boldsymbol{x}_{pbest,G}$ is randomly selected from the top $N \times p$ $(p \in [0, 1])$ members in generation G. The indices r_1, r_2 are randomly selected from $[1, N]$ such that they differ from each other as well as i. The greediness of current-to-$p$best/1 depends on the control parameter p, which trades off exploitation and exploration (small p behaves more greedily).

For each dimension j, if the mutant vector element $v_{j,i,G}$ is outside the search range boundaries $[x_j^{min}, x_j^{max}]$, we applied the same correction performed in [5]:

$$v_{j,i,G} = \begin{cases} (x_j^{min} + x_{j,i,G})/2 & \text{if } v_{j,i,G} < x_j^{min} \\ (x_j^{max} + x_{j,i,G})/2 & \text{if } v_{j,i,G} > x_j^{max} \end{cases} \quad (4)$$

After generating the mutant vector $\boldsymbol{v}_{i,G}$, it is crossed with the parent $\boldsymbol{x}_{i,G}$ in order to generate trial vector $\boldsymbol{u}_{i,G}$. In SHADE 1.1, Binomial Crossover, which is the most commonly used crossover operator in DE, is used and implemented as follows:

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{if } \mathrm{rand}[0, 1) \leq CR_i \text{ or } j = j_{rand} \\ x_{j,i,G} & \text{otherwise} \end{cases} \quad (5)$$

$\mathrm{rand}[0, 1)$ denotes a uniformly selected random number from $[0, 1)$, and j_{rand} is a decision variable index which is uniformly randomly selected from $[1, D]$.

C. Survival for next generation

After all of the trial vectors $\boldsymbol{u}_{i,G}, 0 \leq i \leq N$ have been generated, a selection process determines the survivors for the next generation. The selection operator in standard DE compares each individual $\boldsymbol{x}_{i,G}$ against its corresponding trial vector $\boldsymbol{u}_{i,G}$, keeping the better vector in the population.

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} & \text{if } f(\boldsymbol{u}_{i,G}) \leq f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{cases} \quad (6)$$

D. External archive

In JADE, an optional, *external archive* is used for maintaining diversity [5]. SHADE 1.1 also adopts an external archive. Parent vectors $\boldsymbol{x}_{i,G}$ which were worse than the trial vectors $\boldsymbol{u}_{i,G}$ (and are not selected for survival in the standard DE, Eq. (6)) are preserved. When the archive is used, $\boldsymbol{x}_{r2,G}$ in Eq. (3) is selected from $\boldsymbol{P} \cup \boldsymbol{A}$, the union of the population $\boldsymbol{P}$ and the archive $\boldsymbol{A}$. Whenever the size of the archive exceeds the predefined archive size $|\boldsymbol{A}|$, randomly selected elements are deleted to make space for the newly inserted elements.

E. Historical-memory update

In each generation, in Eq. (6), CR_i and F_i values that succeed in generating a trial vector $\boldsymbol{u}_{i,G}$ which is better than the parent individual $\boldsymbol{x}_{i,G}$ are recorded as S_{CR}, S_F, and at the end of the generation, the memory contents are updated using Algorithm 1.

Algorithm 1: Memory update algorithm in SHADE 1.1

if $S_{CR} \neq \emptyset$ **and** $S_F \neq \emptyset$ **then**
  if $M_{CR,k,G} = \perp$ **or** $\max(S_{CR}) = 0$ **then**
    $M_{CR,k,G+1} = \perp$;
  else
    $M_{CR,k,G+1} = \text{mean}_{WL}(S_{CR})$;
  $M_{F,k,G+1} = \text{mean}_{WL}(S_F)$;
  $k++$;
  If $k > H$, $k = 1$;
else
  $M_{CR,k,G+1} = M_{CR,k,G}$;
  $M_{F,k,G+1} = M_{F,k,G}$;

In Algorithm 1, index k ($1 \leq k \leq H$) determines the position in the memory to update. In generation G, the k-th element in the memory is updated. At the beginning of the search k is initialized to 1. k is incremented whenever a new element is inserted into the history. If $k > H$, k is set to 1. In the update algorithm 1, note that when all individuals in generation G fail to generate a trial vector which is better than the parent, i.e., $S_{CR} = S_F = \emptyset$, the memory is not updated.

The weighted Lehmer mean $\text{mean}_{WL}(S)$ is computed using the formula below, and as with [21], the amount of fitness improvement Δf_k is used in order to influence the parameter adaptation (S refers to either S_{CR} or S_F).

$$\text{mean}_{WL}(S) = \frac{\sum_{k=1}^{|S|} w_k \cdot S_k^2}{\sum_{k=1}^{|S|} w_k \cdot S_k} \tag{7}$$

$$w_k = \frac{\Delta f_k}{\sum_{l=1}^{|S_{CR}|} \Delta f_l} \tag{8}$$

$$\Delta f_k = |f(\boldsymbol{u}_{k,G}) - f(\boldsymbol{x}_{k,G})| \tag{9}$$

As M_{CR} is updated, if $M_{CR,k,G} = \perp$ (where $\perp$ denotes a special, "terminal value") or $\max(S_{CR}) = 0$ (i.e., all elements of S_{CR} are 0), $M_{CR,k,G+1}$ is set to $\perp$. Thus, if M_{CR} is assigned the terminal value $\perp$, then M_{CR} will remain fixed at $\perp$ until the end of the search. This has the effect of locking CR_i to 0 until the end of the search, causing the algorithm to enforce a "change-one-parameter-at-a-time" policy, which tends to slow down convergence, and is effective on multimodal problems.

F. Linear Population Size Reduction (LPSR) and L-SHADE algorithm

As mentioned in Section I, population size reduction has been shown to be highly effective in improving EA performance [15], [16]. In order to improve the performance of SHADE 1.1, we incorporate a population size reduction method for dynamically resizing the population during a DE run. Laredo et al proposed SVPS [16], a general framework for

Algorithm 2: L-SHADE algorithm

// Initialization phase
$G = 1$, $N_G = N^{init}$, Archive $\boldsymbol{A} = \emptyset$;
Initialize population $\boldsymbol{P}_G = (\boldsymbol{x}_{1,G}, ..., \boldsymbol{x}_{N,G})$ randomly;
Set all values in M_{CR}, M_F to 0.5;
// Main loop
while *The termination criteria are not met* **do**
  $S_{CR} = \emptyset, S_F = \emptyset$;
  for $i = 1$ **to** N **do**
    r_i = Select from $[1, H]$ randomly;
    If $M_{CR,r_i} = \perp$, $CR_{i,G} = 0$. Otherwise $CR_{i,G} = \text{randn}_i(M_{CR,r_i}, 0.1)$;
    $F_{i,G} = \text{randc}_i(M_{F,r_i}, 0.1)$;
    Generate trial vector $\boldsymbol{u}_{i,G}$ according to current-to-$p$best/1/bin;
  for $i = 1$ **to** N **do**
    if $f(\boldsymbol{u}_{i,G}) \leq f(\boldsymbol{x}_{i,G})$ **then**
      $\boldsymbol{x}_{i,G+1} = \boldsymbol{u}_{i,G}$;
    else
      $\boldsymbol{x}_{i,G+1} = \boldsymbol{x}_{i,G}$;
    if $f(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G})$ **then**
      $\boldsymbol{x}_{i,G} \rightarrow \boldsymbol{A}$;
      $CR_{i,G} \rightarrow S_{CR}$, $F_{i,G} \rightarrow S_F$;
  If necessary, delete randomly selected individuals from the archive such that the archive size is $|\boldsymbol{A}|$.
  Update memories M_{CR} and M_F (Algorithm 1);
  // Optional LPSR strategy
  Calculate N_{G+1} according to Eq. (10);
  if $N_G < N_{G+1}$ **then**
    Sort individuals in $\boldsymbol{P}$ based on their fitness values and delete lowest $N_G - N_{G+1}$ members;
    Resize archive size $|\boldsymbol{A}|$ according to new $|\boldsymbol{P}|$;
  G++;

flexibly defining a population reduction schedule. We use Linear Population Size Reduction (LPSR), a simple specialization of SVPS which reduces the population linearly as a function of the number of fitness evaluations. LPSR continuously reduces the population to match a linear function where the population size at generation 1 is N^{init}, and the population at the end of the run is N^{min}. After each generation G, the population size in the next generation, N_{G+1}, is computed according to the formula:[1]

$$N_{G+1} = \text{round}\left[\left(\frac{N^{min} - N^{init}}{MAX\_NFE}\right) \cdot NFE + N^{init}\right] \tag{10}$$

N^{min} is set to the smallest possible value such that the evolutionary operators can be applied – in the case of L-SHADE, $N^{min} = 4$ because the current-to-$p$best mutation operator Eq. (3) requires 4 individuals. NFE is the current number of fitness evaluations, and MAX_NFE is the maximum number of fitness evaluations. Whenever $N_{G+1} < N_G$, the $(N_G - N_{G+1})$ worst-ranking individuals are deleted from the population.

[1] Eq. (10) is almost equivalent to SVPS when the SVPS control parameters are set to $\tau = 1, \rho = 0$. See [16] for a detailed description of SVPS.

TABLE I: The results of L-SHADE on the CEC2014 benchmarks for $D = 10, 30, 50$ and 100 dimensions. Each column shows best, worst, median, mean and standard deviation of the error value between the best fitness values found in each run and the true optimal value. The maximum number of objective function evaluations is $D \times 10,000$. All results are based on 51 runs.

	$D = 10$					$D = 30$					$D = 50$					$D = 100$				
Func.	Best	Worst	Median	Mean	Std.	Best	Worst	Median	Mean	Std.	Best	Worst	Median	Mean	Std.	Best	Worst	Median	Mean	Std.
1	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	2.9e+01	9.5e+03	8.2e+02	1.2e+03	1.5e+03	8.0e+04	3.4e+05	1.6e+05	1.7e+05	5.7e+04
2	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00
3	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00
4	0.0e+00	3.5e+01	3.5e+01	2.9e+01	1.3e+01	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	7.2e-01	9.8e+01	9.8e+01	5.9e+01	4.6e+01	1.0e+02	2.1e+02	1.5e+02	1.7e+02	3.1e+01
5	1.5e-01	2.0e+01	2.0e+01	1.4e+01	8.8e+00	2.0e+01	2.0e+01	2.0e+01	2.0e+01	3.7e-02	2.0e+01	2.0e+01	2.0e+01	2.0e+01	4.6e-02	2.0e+01	2.1e+01	2.1e+01	2.1e+01	3.1e-02
6	0.0e+00	8.9e-01	0.0e+00	1.8e-02	1.3e-01	0.0e+00	7.1e-06	0.0e+00	1.4e-07	9.9e-07	4.9e-05	1.8e+00	1.8e-02	2.6e-01	5.2e-01	4.5e+00	1.4e+01	8.8e+00	8.7e+00	2.3e+00
7	0.0e+00	2.5e-02	0.0e+00	3.0e-03	6.5e-03	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00
8	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	3.3e-08	0.0e+00	2.6e-09	7.5e-09	2.4e-03	3.7e-02	9.6e-03	1.1e-02	7.4e-03
9	2.2e-03	5.0e+00	2.0e+00	2.3e+00	8.4e-01	3.3e+00	9.2e+00	7.1e+00	6.8e+00	1.5e+00	5.4e+00	1.5e+01	1.1e+01	1.1e+01	2.1e+00	2.4e+01	4.6e+01	3.4e+01	3.4e+01	5.0e+00
10	0.0e+00	6.2e-02	0.0e+00	8.6e-03	2.2e-02	0.0e+00	6.2e-02	2.1e-02	1.6e-02	1.6e-02	5.4e-02	2.3e-01	1.1e-01	1.2e-01	4.1e-02	1.6e+01	4.2e+01	2.6e+01	2.6e+01	5.8e+00
11	3.9e-01	1.4e+02	1.6e+01	3.2e+01	3.8e+01	8.5e+02	1.7e+03	1.2e+03	1.2e+03	1.8e+02	2.5e+03	3.8e+03	3.3e+03	3.2e+03	3.3e+02	9.2e+03	1.2e+04	1.1e+04	1.1e+04	5.6e+02
12	2.5e-02	1.2e-01	7.0e-02	6.8e-02	1.9e-02	1.1e-01	2.3e-01	1.6e-01	1.6e-01	2.3e-02	1.5e-01	2.7e-01	2.2e-01	2.2e-01	2.8e-02	3.2e-01	5.2e-01	4.4e-01	4.4e-01	4.7e-02
13	1.6e-02	8.7e-02	5.3e-02	5.2e-02	1.5e-02	9.0e-02	1.7e-01	1.2e-01	1.2e-01	1.7e-02	1.1e-01	2.1e-01	1.6e-01	1.6e-01	1.8e-02	1.9e-01	2.9e-01	2.4e-01	2.4e-01	2.1e-02
14	4.5e-02	1.6e-01	7.6e-02	8.1e-02	2.6e-02	1.8e-01	3.0e-01	2.4e-01	2.4e-01	3.0e-02	2.4e-01	3.5e-01	2.9e-01	3.0e-01	2.5e-02	1.0e-01	1.3e-01	1.2e-01	1.2e-01	7.3e-03
15	2.1e-01	5.2e-01	3.7e-01	3.7e-01	6.9e-02	1.6e+00	2.6e+00	2.1e+00	2.1e+00	2.5e-01	4.1e+00	6.2e+00	5.1e+00	5.2e+00	5.1e-01	1.4e+01	1.9e+01	1.6e+01	1.6e+01	1.2e+00
16	4.0e-01	1.7e+00	1.3e+00	1.2e+00	3.0e-01	7.2e+00	9.5e+00	8.6e+00	8.5e+00	4.6e-01	1.6e+01	1.8e+01	1.7e+01	1.7e+01	4.8e-01	3.8e+01	4.0e+01	3.9e+01	3.9e+01	4.8e-01
17	0.0e+00	5.6e+00	6.2e-01	9.8e-01	1.1e+00	4.9e+01	3.3e+02	2.0e+02	1.9e+02	7.5e+01	4.9e+02	2.6e+03	1.2e+03	1.4e+03	5.1e+02	2.5e+03	5.9e+03	4.4e+03	4.4e+03	7.1e+02
18	4.3e-03	1.5e+00	1.6e-01	2.4e-01	3.1e-01	3.5e-01	1.7e+01	5.3e+00	5.9e+00	2.9e+00	6.8e+01	1.4e+02	9.5e+01	9.7e+01	1.4e+01	1.9e+02	2.6e+02	2.3e+02	2.2e+02	1.7e+01
19	1.3e-02	3.8e-01	6.2e-02	7.7e-02	6.4e-02	1.6e+00	4.9e+00	3.9e+00	3.7e+00	6.8e-01	5.4e+00	1.2e+01	7.9e+00	8.3e+00	1.8e+00	9.1e+01	1.0e+02	9.5e+01	9.6e+01	2.3e+00
20	1.4e-02	6.1e-01	9.5e-02	1.8e-01	1.8e-01	8.8e-01	7.6e+00	3.1e+00	3.1e+00	1.5e+00	6.1e+00	2.5e+01	1.4e+01	1.4e+01	4.6e+00	7.2e+01	2.5e+02	1.4e+02	1.5e+02	5.2e+01
21	3.5e-04	1.1e+00	4.3e-01	4.1e-01	3.1e-01	1.8e+00	3.7e+02	3.2e+01	8.7e+01	9.0e+01	2.5e+02	1.0e+03	5.0e+02	5.2e+02	1.5e+02	1.1e+03	3.8e+03	2.2e+03	2.3e+03	5.3e+02
22	3.6e-04	1.1e-01	4.4e-02	4.4e-02	2.8e-02	9.7e+00	1.4e+02	2.5e+01	2.8e+01	1.8e+01	3.2e+01	2.8e+02	6.9e+01	1.1e+02	7.5e+01	6.1e+02	1.4e+03	1.1e+03	1.1e+03	1.9e+02
23	3.3e+02	3.3e+02	3.3e+02	3.3e+02	0.0e+00	3.2e+02	3.2e+02	3.2e+02	3.2e+02	0.0e+00	3.4e+02	3.4e+02	3.4e+02	3.4e+02	4.4e-13	3.5e+02	3.5e+02	3.5e+02	3.5e+02	2.8e-13
24	1.0e+02	1.1e+02	1.1e+02	1.1e+02	2.3e+00	2.2e+02	2.3e+02	2.2e+02	2.2e+02	1.1e+00	2.7e+02	2.8e+02	2.8e+02	2.8e+02	6.6e-01	3.9e+02	4.0e+02	3.9e+02	3.9e+02	2.9e+00
25	1.0e+02	2.0e+02	1.1e+02	1.3e+02	4.0e+01	2.0e+02	2.0e+02	2.0e+02	2.0e+02	5.0e-02	2.0e+02	2.1e+02	2.1e+02	2.1e+02	3.6e-01	2.0e+02	2.0e+02	2.0e+02	2.0e+02	4.0e-13
26	1.0e+02	1.0e+02	1.0e+02	1.0e+02	1.6e-02	1.0e+02	1.0e+02	1.0e+02	1.0e+02	1.6e-02	1.0e+02	2.0e+02	1.0e+02	1.0e+02	1.4e+01	2.0e+02	2.0e+02	2.0e+02	2.0e+02	6.2e-13
27	8.5e-01	4.0e+02	1.5e+00	5.8e+01	1.3e+02	3.0e+02	3.0e+02	3.0e+02	3.0e+02	0.0e+00	3.0e+02	3.9e+02	3.3e+02	3.3e+02	3.0e+01	3.3e+02	4.6e+02	3.7e+02	3.8e+02	3.3e+01
28	3.6e+02	4.7e+02	3.7e+02	3.8e+02	3.2e+01	8.1e+02	8.7e+02	8.4e+02	8.4e+02	1.4e+01	1.1e+03	1.2e+03	1.1e+03	1.1e+03	2.9e+01	2.2e+03	2.4e+03	2.3e+03	2.3e+03	4.6e+01
29	2.2e+02	2.2e+02	2.2e+02	2.2e+02	4.6e-01	7.1e+02	7.4e+02	7.1e+02	7.2e+02	5.1e+00	7.4e+02	9.0e+02	7.9e+02	7.9e+02	2.4e+01	7.0e+02	1.1e+03	7.9e+02	8.0e+02	7.6e+01
30	4.6e+02	5.5e+02	4.6e+02	4.6e+02	1.3e+01	4.8e+02	3.5e+03	1.1e+03	1.2e+03	6.2e+02	7.9e+03	1.0e+04	8.6e+03	8.7e+03	4.1e+02	5.7e+03	1.0e+04	8.3e+03	8.3e+03	9.6e+02

Finally, the overall L-SHADE algorithm is shown in Algorithm 2. Lines 21–24 implement the LPSR population reduction scheme used by L-SHADE. If this option is excluded, Algorithm 2 becomes SHADE 1.1 algorithm. Note that in line 24, the archive size $|\boldsymbol{A}|$ is readjusted, because we found in preliminary experiments that this resulted in better performance than setting the archive size to a constant value $|\boldsymbol{A}| = N^{init}$.

III. Evaluating L-SHADE on the CEC2014 benchmarks

This section presents an empirical evaluation of L-SHADE. We evaluated the performance of L-SHADE on the *CEC2014 Special Session on Real-Parameter Single Objective Optimization* benchmark suite [17] and compared L-SHADE to both state-of-the-art DE methods, as well as state-of-the-art restart CMA-ES methods.

The CEC2014 benchmark set consists of 30 test functions. For all of the problems, the search space is $[-100, 100]^D$.[2] Functions $F_1 \sim F_3$ are unimodal. $F_4 \sim F_{16}$ are simple multimodal functions. $F_{17} \sim F_{22}$ are hybrid functions. Finally, $F_{23} \sim F_{30}$ are composite functions which combine multiple test problems into a complex landscape. See [17] for details.

[2]We treat the outside of this search space $[-100, 100]^D$ as infeasible. Note that local optima outside this search space are easily discovered, and have a better objective function values than local optima solutions inside $[-100, 100]^D$ on some functions (in particular, composite functions $F_{23} \sim F_{30}$). This issue has also been found in the CEC2005 benchmarks [22].

The main difference between the CEC 2014 benchmarks and the previous CEC 2005/2013 benchmarks [23], [24] is the inclusion of a new set of *hybrid functions* ($F_{17} \sim F_{22}$). In these hybrid functions, the variables in the solution vector are randomly partitioned into $3 \sim 5$ groups, and each group is evaluated using a different function, each with a distinct structure. This is a kind of partial separability, which is present in real world problems such as transportation networks and circuit theory, image processing, etc. [25]. Tang et. al. pointed out that most previous benchmark sets did not include partially separable functions, and consisted of functions where all variables have uniform features [26]. Taking this into consideration, the CEC2014 benchmarks adopted hybrid functions ($F_{17} \sim F_{22}$) in order to more closely approximate real-world benchmarks.

We performed our evaluation following the guidelines of the CEC2014 benchmark competition [17]. When the gap between the values of the best solution found and the optimal solution was 10^{-8} or smaller, the error (score) was treated as 0. For all of the problems the number of dimensions $D = 10, 30, 50, 100$, and the maximum number of objective function calls per run was $D \times 10,000$ (i.e., 100,000, 300,000, 500,000 and 1,000,000 respectively). The number of runs per problem was 51, and the average performance of these runs was evaluated.

The results of L-SHADE on $D = 10, 30, 50$ and 100 dimensions are shown in Table I. Each column shows best, worst, median, mean and standard deviation of the error value between the best fitness values found in each run and the true optimal value.

TABLE II: The 4 control parameters of L-SHADE.

Parameters	Ranges	Default settings	Tuned settings
$r^{N^{init}}$	{15, 16, ..., 24, 25}	20	18
r^{arc}	{1.0, 1.1, ..., 2.9, 3.0}	2.0	2.6
p	{0.05, 0.06, ..., 0.14, 0.15}	0.1	0.11
H	{2, 3, ..., 9, 10}	5	6

TABLE III: Algorithm Complexity

	$T0$	$T1$	$\hat{T}2$	$(\hat{T}2 - T1)/T0$
$D = 10$	0.11	0.21	0.38	1.53
$D = 30$		0.97	1.41	3.95
$D = 50$		2.37	3.04	5.91
$D = 100$		8.71	10.09	12.25

A. Algorithm Parameters

This section describes the parameter setting of L-SHADE and its execution environment. In this paper, for tuning the parameter settings of L-SHADE, we use ParamILS [27] which is a versatile and efficient automatic parameter tuner which can be applied to a wide range of parameterized algorithms, including optimization algorithms. We used ParamILS because it has been shown to be highly successful in tuning search and optimization algorithms [27], and ParamILS is significantly less labor-intensive compared to manual tuning.

The control parameters of L-SHADE are: (1) initial population size N^{init}, (2) external archive size $|\boldsymbol{A}|$, (3) historical memory size H, (4) p value for current-to-$p$best/1 mutation (Eq. (3)). Following the standard practice for setting the population size in DE [1], we set the initial population size N^{init} to the dimensionality D of the functions multiplied by a parameter $r^{N^{init}}$, i.e., $N^{init} = \mathrm{round}(D \times r^{N^{init}})$. Similarly, the external archive size $|\boldsymbol{A}|$ is set to N^{init} multiplied by a parameter r^{arc}, i.e., $|\boldsymbol{A}| = \mathrm{round}(N^{init} \times r^{arc})$. We used the default control parameters for the ParamILS parameter tuner. The max number of parameter configurations evaluated by ParamILS (i.e. the number of execution times of L-SHADE) was set to $1,000$. The objective function used by ParamILS was the error (difference) between the best fitness values found in L-SHADE's run and the true optimal value. Following [20], we tried to avoid overfitting the control parameters to the CEC2014 benchmarks by using 5 composition functions $F_{21} \sim F_{25}$ from the CEC2013 benchmarks [24] as the training problem set for the ParamILS parameter tuner. Note that these training functions are not from the CEC2014 benchmarks (i.e., the test problem set). Dimension sizes are set to 10 and 30 (i.e. the total number of training problems are $5 \times 2 = 10$). Execution of L-SHADE algorithm in this tuning phase was according to the CEC2013 benchmark competition rules, which are identical with the CEC2014 competition rules described above.

Table II shows the search range of control parameters, the default parameter settings (i.e. initial parameter configuration of ParamILS), and the best settings found by ParamILS, which are used as the L-SHADE's parameter settings in the remainder of this paper. The default parameter settings were based on preliminary experiments. The parameter tuning process using ParamILS executed for approximately 4 hours. Due to space constraints, we do not show the comparison results, but L-SHADE with the tuned parameter settings by ParamILS is slightly better than one with the default parameter settings.

B. Algorithm Complexity

This section describes the algorithm complexity of our L-SHADE code as defined in [17]. All experiments were executed on the following system:

- OS: Ubuntu 12.04 LTS
- CPU: core i7 (2.20GHz)
- RAM: 8GB
- Language: C++
- Compiler: g++ (gcc) with -O3 optimization flag

Table III shows the computed algorithm complexity on 10, 30, 50 and 100 dimensions. As defined in [24], $T0$ is the time calculated by running the following test problem:

```
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
```

$T1$ is the time to execute $200,000$ evaluations of benchmark function F_{18} by itself with D dimensions, and $T2$ is the time to execute SHADE with $200,000$ evaluations of F_{18} in D dimensions. $\hat{T}2$ is the mean $T2$ values of 5 runs. According to Table III, both $T1$ and $\hat{T}2$ scaled linearly with the number of dimensions, as shown by the linear growth of $(\hat{T}2 - T1)/T0$.

C. L-SHADE vs. state-of-the-art DE algorithms on the CEC2014 benchmarks

We compared L-SHADE with the state-of-the-art DE algorithms SHADE 1.1 [9], CoDE [18], EPSDE [6], SaDE [4], JADE [5], and dynNP-jDE [15] on the CEC2014 benchmarks. The Matlab source code for CoDE, EPSDE, SaDE and JADE were downloaded from [28], the web site of Q. Zhang, one of the authors of [18]. These were used for the experiments in [18], and are based on code originally received from the original authors of CoDE, EPSDE, SaDE and JADE. We minimally modified these programs so that they would work with the CEC2014 benchmark codes. We implemented the programs of SHADE 1.1 and dynNP-jDE by C++ and Eq. (4) is applied as the bound handling method. For each algorithm, we used the control parameter values that were suggested in the cited, original papers.

The results for 10, 30, 50 and 100 dimensions are shown in Tables IV. Due to space constraints, we only show the aggregate results of statistical testing $(+, -, \approx)$. In Tables IV, we show the aggregate results of comparing each algorithm vs. L-SHADE on 30 functions. For example, according to

TABLE IV: Comparison of L-SHADE with state-of-the-art DE algorithms on the CEC2014 benchmarks ($D = 10, 30, 50, 100$ dimensions). The aggregate results of statistical testing $(+, -, \approx)$ on 30 functions are shown. The symbols $+, -, \approx$ indicate that a given algorithm performed significantly better $(+)$, significantly worse $(-)$, or not significantly different better or worse $(\approx)$ compared to L-SHADE using the Wilcoxon rank-sum test (significantly, $p < 0.05$). The maximum number of objective function evaluations is $D \times 10,000$. All results are based on 51 runs.

vs. L-SHADE		$D = 10$	$D = 30$	$D = 50$	$D = 100$
SHADE	+ (better)	0	3	5	7
	− (worse)	16	18	19	17
	≈ (no sig.)	14	9	6	6
CoDE	+ (better)	6	4	6	4
	− (worse)	12	19	23	22
	≈ (no sig.)	12	7	1	4
EPSDE	+ (better)	4	5	6	7
	− (worse)	20	22	21	23
	≈ (no sig.)	6	3	3	0
JADE	+ (better)	1	2	3	6
	− (worse)	20	22	23	20
	≈ (no sig.)	9	6	4	4
SaDE	+ (better)	4	1	0	1
	− (worse)	17	25	27	27
	≈ (no sig.)	9	4	3	2
dynNP-jDE	+ (better)	6	3	5	7
	− (worse)	16	20	20	18
	≈ (no sig.)	8	7	5	5

the Wilcoxon test on the column $D = 10$ of Table IV, which summarizes the experimental results on $D = 10$, CoDE performed (significantly, $p < 0.05$) better than L-SHADE on 6 functions and worse than L-SHADE on 12 functions.

As shown in the tables counting the number of $+$, $-$, and $\approx$ results, L-SHADE clearly has the best overall performance on these 30 problems for all $D \in \{10, 30, 50\}$. These results shows that L-SHADE, the proposed methods, significantly outperform previous state-of-the-art DE variants overall on the CEC2014 benchmarks. This was to be expected since our previous work [7] shows that SHADE can also perform better than these methods. Also, note that the search performance of L-SHADE is significantly better than SHADE 1.1, showing that LPSR successfully contributed to L-SHADE's performance.

D. Comparing L-SHADE with state-of-the-art restart CMA-ES variants

Currently, restart CMA-ES methods based on IPOP-CMA-ES [12] are considered the state-of-the-art methods for single objective, real parameter optimization problems – restart CMA-ES variants have won the CEC2005 and 2013 competitions [23], [24], as well as the GECCO2009/2010/2012/2013 BBOB [29]. On the other hand, it is widely believed that DE methods perform significantly worse than restart CMA-ES [2]. Our results in Section III-C indicate that the performance of L-SHADE is quite competitive compared to previous DE algorithms. A natural question is whether L-SHADE, which seems to establish a new state-of-the-art for DE, is competitive with restart CMA-ES. Thus, this section presents an comparison of L-SHADE with state-of-the-art restart CMA-ES variants.

The evaluation includes NBIPOP-$_{\mathrm{A}}$CMA-ES [19] and iCMAES-ILS [20], sophisticated restart CMA-ES variants based on IPOP-CMA-ES [12], which tied for first place in the recent *Special Session & Competition on Real-Parameter Single Objective Optimization* held at CEC-2013. NBIPOP-$_{\mathrm{A}}$CMA-ES [19], which is improved variants of BIPOP-CMA-ES [30], adaptively allocates resources among the two components (IPOP-CMA-ES and multistart CMA-ES). iCMAES-ILS [20] is a hybrid method which combines IPOP-CMA-ES and an iterated local search method. iCMAES-ILS separates the search in two components, and initially allocates computational resources evenly between the two. After some time has passed, the best-so-far solutions found by these components is compared, and the better method is assigned all of the remaining time. Both algorithms have several characteristics in common: (1) they are based on CMA-ES, (2) restarts are performed, (3) compared to L-SHADE, the algorithms are extremely complex. iCMAES-ILS and NBIPOP-$_{\mathrm{A}}$CMA-ES allocate computational resources among multiple search methods (and multiple control parameter sets) in an effort to make the overall algorithm more robust and avoid the risk of failure, i.e., these can be seen as instances of an algorithm portfolio approach [31]. All of these features are in direct contrast to L-SHADE.

In addition to an overall comparison, it is interesting to compare L-SHADE with CMA-ES variants on specific subclasses of problems in the CEC2014 benchmark set, especially the new hybrid functions. We classified the 30 CEC2014 benchmarks into 4 groups: 3 unimodal functions, 13 simple multimodal functions, 6 hybrid functions, 8 composition functions, and evaluated the performance on both groups separately. The results for $D \in \{10, 30, 50, 100\}$ dimensions on these 4 groups and the overall results on all 30 functions are shown in Table V. Due to space, the table only shows the aggregate results of comparing each algorithm vs. L-SHADE using the Wilcoxon rank-sum test (significantly, $p < 0.05$). The source code for iCMAES-ILS and NBIPOP-$_{\mathrm{A}}$CMA-ES[3], were downloaded from [32], the web site of CEC2013 competitions. For L-SHADE, we use the data from Section III-C.

We observe the following: On 3 unimodal functions, L-SHADE performs similarly to NBIPOP-$_{\mathrm{A}}$CMA-ES and iCMAES-ILS for 10, 30 dimensions, but L-SHADE is outperformed by both methods on one function (F_1) for 50, 100 dimensions. On the 13 simple, multimodal functions, L-SHADE performs better than NBIPOP-$_{\mathrm{A}}$CMA-ES on 10, 30 dimensions and comparably to iCMAES-ILS on 10 dimensions, but both methods outperform L-SHADE on higher dimension size. On the 6 hybrid functions, L-SHADE clearly outperforms NBIPOP-$_{\mathrm{A}}$CMA-ES and iCMAES-ILS for all dimensions. This results indicate that L-SHADE is able to outperform state-of-the-art restart CMA-ES variants on real-world problem which variables in the solution vector might have non-uniform features. On the 8 composition functions, L-SHADE outperforms NBIPOP-$_{\mathrm{A}}$CMA-ES for 30, 50, 100 dimensions. While L-SHADE performs better than iCMAES-ILS for 10, 50 dimensions, L-SHADE is worse for 30, 100 dimensions. Overall on these 30 benchmark functions, L-SHADE performs better than NBIPOP-$_{\mathrm{A}}$CMA-ES on 10, 30, 50 dimensions,

[3]Since the search range in the program code of NBIPOP-$_{\mathrm{A}}$CMA-ES is set to $[-\infty, \infty]^D$, we changed this range to $[-100, 100]^D$, to adhere to the CEC2014 competition rules.

TABLE V: Comparison of L-SHADE with NBIPOP-$_{\text{A}}$CMA-ES and iCMAES-ILS on the CEC2014 benchmarks for $D \in \{10, 30, 50, 100\}$ on 4 groups (3 unimodal functions $F_1 \sim F_3$ (first column), 13 simple multimodal functions $F_4 \sim F_{16}$ (second column), 6 hybrid functions $F_{17} \sim F_{22}$ (third column), 8 composition functions $F_{23} \sim F_{30}$ (fourth column)) and the overall results on all 30 functions (final column).

Groups	vs. L-SHADE (Wilcoxon rank-sum, $p < 0.05$)	NBIPOP-$_{\text{A}}$CMA-ES				iCMAES-ILS			
		$D = 10$	$D = 30$	$D = 50$	$D = 100$	$D = 10$	$D = 30$	$D = 50$	$D = 100$
3 Unimodal Functions	$+$ (better)	0	0	1	1	0	0	1	1
	$-$ (worse)	0	0	0	0	0	0	0	0
	$\approx$ (no sig.)	3	3	2	2	3	3	2	2
13 Simple Multimodal Functions	$+$ (better)	5	4	6	6	6	6	9	7
	$-$ (worse)	7	5	4	4	6	4	3	4
	$\approx$ (no sig.)	1	4	3	3	1	3	1	2
6 Hybrid Functions	$+$ (better)	0	0	0	1	0	0	0	1
	$-$ (worse)	6	6	6	5	6	6	5	4
	$\approx$ (no sig.)	0	0	0	0	0	0	1	1
8 Composition Functions	$+$ (better)	3	4	3	3	5	2	3	3
	$-$ (worse)	3	1	2	2	3	3	2	4
	$\approx$ (no sig.)	2	3	3	3	0	3	3	1
30 All Functions	$+$ (better)	8	8	10	11	11	8	13	12
	$-$ (worse)	16	12	12	11	15	13	10	12
	$\approx$ (no sig.)	6	10	8	8	4	9	7	6

and comparably for 100 dimensions. L-SHADE outperforms iCMAES-ILS on 10, 30 dimensions, worse than iCMAES-ILS for 50 dimensions, and comparably to iCMAES-ILS on 100 dimensions.

Overall, we have demonstrated that contrary to current conventional wisdom (c.f. [2]), we have shown that a DE approach can be quite competitive with state-of-the-art restart CMA-ES variants.

E. Comparison between LPSR and DPSR

This section compares LPSR with DPSR [15], to see which mechanism is better suited as a population resizing mechanism for the SHADE 1.1 framework. We compare L-SHADE with D-SHADE, a variant of SHADE 1.1 which uses DPSR instead of LPSR. In D-SHADE, the linear population size reduction mechanism of L-SHADE for lines 21–24 in Algorithm 2 is replaced by DPSR strategy (See [15]). The parameter settings of D-SHADE is tuned by ParamILS same as L-SHADE. Table VI shows the search range of control parameters, the default parameter settings, and the best settings found by ParamILS, which are used as the D-SHADE's parameter settings. Where, the tuning scenario of ParamILS is same as Section III-A. Interestingly, as shown in Table II and VI, the tuned parameter values of $r^{N^{init}}$, r^{arc} and p are very similar between L-SHADE and D-SHADE.

Table VII shows the aggregate results of the comparison between L-SHADE and D-SHADE for $D = 10, 30, 50$, and 100 dimensions. L-SHADE clearly outperforms D-SHADE for 10 and 30 dimensions. However, L-SHADE performs similarly to D-SHADE for $D = 50$ and is outperformed for 100 dimensions. Thus, the best population reduction strategy for SHADE 1.1 appears to depend on the dimensionality of the problem. A more detailed evaluation of population reduction strategies is an avenue for future work. However, compared with DPSR, LPSR has one definite advantage in that LPSR is very simple and has fewer control parameters which the user must tune.

TABLE VI: The 5 control parameters of D-SHADE.

Parameters	Ranges	Default settings	Tuned settings
$r^{N^{init}}$	{15, 16, ..., 24, 25}	20	18
r^{arc}	{1.0, 1.1, ..., 2.9, 3.0}	2.0	2.5
p	{0.05, 0.06, ..., 0.14, 0.15}	0.1	0.11
H	{2, 3, ..., 9, 10}	5	9
r_{freq}	{4, 5, ..., 9, 10}	4	7

TABLE VII: Comparison of L-SHADE with D-SHADE on the CEC2014 benchmarks.

vs. L-SHADE		$D = 10$	$D = 30$	$D = 50$	$D = 100$
D-SHADE	$+$ (better)	2	3	8	13
	$-$ (worse)	13	12	8	8
	$\approx$ (no sig.)	15	15	14	9

IV. Conclusions

This paper proposes L-SHADE, which extends SHADE 1.1 [9] with Linear Population Size Reduction (LPSR). LPSR is a simplified, special case of Simple Variable Population Sizing [16] which reduces the population linearly, and only requires initial population size and population reduction frequency as user-defined parameters. We evaluated the search performance of L-SHADE on the 30 benchmark functions from the *CEC2014 Special Session on Real-Parameter Single Objective Optimization benchmark suite* [17]. The experiments were performed for 10, 30, 50 and 100 dimensions. The experimental results showed that L-SHADE significantly improves upon the performance of SHADE 1.1, and outperforms previous DE variants, including JADE [5], CoDE [18], EPSDE [6], SaDE [4], and dynNP-jDE [15]. In addition, we compared L-SHADE with NBIPOP-$_{\text{A}}$CMA-ES [19] and iCMAES-ILS [20] which are state-of-the-art restart CMA-ES [12] variants and co-winners of the CEC 2013 competition on real-parameter single objective Optimization. The results showed that L-SHADE is

highly competitive with these CMA-ES variants, especially, on the hybrid functions $F_{17} \sim F_{22}$ where subgroups of variables are associated with functions with different properties.

In spite of the successful results of L-SHADE on $F_{17} \sim F_{22}$ in the CEC2014 benchmarks, our recent study [33] shows that on hybrid functions where the components have significantly different search space characteristics unlike $F_{17} \sim F_{22}$ (e.g., a unimodal-multimodal or separable-nonseparable hybrid), the performance of adaptive DE including SHADE 1.1, JADE and jDE tended to degrade dramatically depending on the fraction of variables allocated to each component in the hybrid function. Therefore, designing adaptive DE algorithms that are effective on hybrid objective functions is a challenging task.

Our results indicate that the simple LPSR method incorporated into L-SHADE is also quite powerful, resulting in significant improvement over SHADE 1.1. However, deterministic population resizing methods such as DPSR, SVPS and LPSR often assume an optimization scenario with the maximum number of fitness evaluations. While this is a standard benchmarking methodology in the literature as well as competitions, real-world applications do not necessarily follow this model. In an "anytime optimization" setting where the best-so-far solution is requested at some a priori unknown time, SHADE 1.1, which does not make any assumptions about the termination condition, may outperform L-SHADE. Evaluation of such alternative settings, as well as application of LPSR to other DE variants, are directions for future work.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grants 2324002 and 25330253.

REFERENCES

[1] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *J. Global Optimiz.*, vol. 11, no. 4, pp. 341–359, 1997.

[2] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," *IEEE Tran. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, 2011.

[3] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems," *IEEE Tran. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, 2006.

[4] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization," *IEEE Tran. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, 2009.

[5] J. Zhang and A. C. Sanderson, "JADE: Adaptive Differential Evolution With Optional External Archive," *IEEE Tran. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, 2009.

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

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

[8] ——, "Evaluating the performance of SHADE on CEC 2013 benchmark problems," in *IEEE CEC*, 2013, pp. 1952–1959.

[9] ——, "Success-History Based Parameter Adaptation for Differential Evolution," *submitted for publication*, 2014.

[10] F. G. Lobo and C. F. Lima, "A Review of Adaptive Population Sizing Schemes in Genetic Algorithms," in *GECCO*, 2005, pp. 228–234.

[11] J. Arabas, Z. Michalewicz, and J. J. Mulawka, "GAVaPS - A Genetic Algorithm with Varying Population Size," in *ICEC*, 1994, pp. 73–78.

[12] A. Auger and N. Hansen, "A Restart CMA Evolution Strategy With Increasing Population Size," in *IEEE CEC*, 2005, pp. 1769–1776.

[13] C. García-Martínez, M. Lozano, F. Herrera, D. Molina, and A. M. Sánchez, "Global and local real-coded genetic algorithms based on parent-centric crossover operators," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1088–1113, 2008.

[14] M. A. M. de Oca, T. Stützle, K. V. den Enden, and M. Dorigo, "Incremental Social Learning in Particle Swarms," *IEEE Trans. on Sys., Man, and Cyber., PartB*, vol. 41, no. 2, pp. 368–384, 2011.

[15] J. Brest and M. S. Maučec, "Population size reduction for the differential evolution algorithm," *Appl. Intell.*, vol. 29, no. 3, pp. 228–247, 2008.

[16] J. L. J. Laredo, C. Fernandes, J. J. M. Guervós, and C. Gagné, "Improving Genetic Algorithms Performance via Deterministic Population Shrinkage," in *GECCO*, 2009, pp. 819–826.

[17] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization," Zhengzhou University and Nanyang Technological University, Tech. Rep., 2013.

[18] Y. Wang, Z. Cai, and Q. Zhang, "Differential Evolution With Composite Trial Vector Generation Strategies and Control Parameters," *IEEE Tran. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, 2011.

[19] I. Loshchilov, "CMA-ES with Restarts for Solving CEC 2013 Benchmark Problems," in *IEEE CEC*, 2013, pp. 369–376.

[20] T. Liao and T. Stützle, "Benchmark Results for a Simple Hybrid Algorithm on the CEC 2013 Benchmark Set for Real-parameter Optimization," in *IEEE CEC*, 2013, pp. 1938–1944.

[21] F. Peng, K. Tang, G. Chen, and X. Yao, "Multi-start JADE with knowledge transfer for numerical optimization," in *IEEE CEC*, 2009, pp. 1889–1895.

[22] T. Liao, D. Molina, M. M. A. Montes de Oca, and T. Stützle, "A Note on Bound Constraints Handling for the IEEE CEC05 Benchmark Function Suite," *Evol. Comput. (in press)*, 2014.

[23] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," Nanyang Technological University, Tech. Rep., 2005.

[24] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernández-Díaz, "Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization," Nanyang Technological University, Tech. Rep., 2013.

[25] A. Griewank and P. L. Toint, "On the Unconstrained Optimization of Partially Separable Functions," in *Nonlinear Optimization 1981*. Academic Press, 1982, pp. 301–312.

[26] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization," University of Science and Technology of China, Tech. Rep., 2010.

[27] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle, "ParamILS: An Automatic Algorithm Configuration Framework," *J. Artif. Intell. Res. (JAIR)*, vol. 36, pp. 267–306, 2009.

[28] Zhang's-Website, "http://dces.essex.ac.uk/staff/qzhang."

[29] BBOB, "http://coco.gforge.inria.fr/doku.php."

[30] N. Hansen, "Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed," in *GECCO (Companion)*, 2009, pp. 2389–2396.

[31] B. Huberman, R. Lukose, and T. Hogg, "An Economics Approach to Hard Computational Problems," *Science*, vol. 275, no. 5296, pp. 51–54, 1997.

[32] CEC2013, "http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2013/CEC2013.htm."

[33] R. Tanabe and A. Fukunaga, "On the Pathological Behavior of Adaptive Differential Evolution on Hybrid Objective Functions," in *GECCO (accepted)*, 2014.