2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Success-History Based Parameter Adaptation for Differential Evolution

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

***Abstract*—Differential Evolution is a simple, but effective approach for numerical optimization. Since the search efficiency of DE depends significantly on its control parameter settings, there has been much recent work on developing self-adaptive mechanisms for DE. We propose a new, parameter adaptation technique for DE which uses a historical memory of successful control parameter settings to guide the selection of future control parameter values. The proposed method is evaluated by comparison on 28 problems from the CEC2013 benchmark set, as well as CEC2005 benchmarks and the set of 13 classical benchmark problems. The experimental results show that a DE using our success-history based parameter adaptation method is competitive with the state-of-the-art DE algorithms.**

I. Introduction

Differential Evolution (DE) is a stochastic search method, that was primarily designed for numerical optimization problems [1]. Despite of its relative simplicity, DE has been shown to be competitive with other more complex optimization algorithms, and has been applied to many practical problems [2].

As with other evolutionary algorithms, the search performance of DE algorithms depends on control parameter settings. A standard DE has three main control parameters, which are the population size, scaling factor F, and crossover rate CR. However, 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, self-adaptive mechanisms for adjusting the control parameters on-line during the search process have been studied by many researchers [3], [2].

JADE [4] is a well-known, effective DE variant which employs a control parameter adaptation mechanism. In addition to on-line, parameter adaptation, JADE also uses a novel mutation strategy called current-to-$p$best/1 and an external archive for storing previously generated individuals. Instead of a static crossover rate CR and scaling factor F, JADE has two corresponding, adaptive variables, μ_{CR}, μ_F. The crossover rate and scaling factor associated with each individual are generated according to a normal/Cauchy distribution with means μ_{CR}, μ_F. At the end of each generation, the values of μ_{CR}, μ_F are updated according to the CR, F pair that resulted in the generation of the successful trial vector in that generation. As the search progresses, μ_{CR}, μ_F should gradually approach the optimal values for the given problem.

In this paper, we propose Success-History based Adaptive DE (SHADE), an enhancement to JADE which uses a history based parameter adaptation scheme. Instead of generating new control parameter settings based on some distribution around a single pair of parameters μ_{CR}, μ_F, we use a *historical memory* M_{CR}, M_F which stores a sets 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. We show experimentally that SHADE outperforms previous DE variants, including JADE, CoDE [5], EPSDE [6], and dynNP-jDE [7].

The rest of the paper is organized as follows. In Section II, we review the basic DE algorithm. The recent work on adaptive DE variants is described in Section III. Section IV reviews JADE, the adaptive DE which is extended by this work. We propose SHADE, a new algorithm using a history based adaptation mechanism in Section V. Section VI presents an empirical evaluation of SHADE to several state-of-the-art DE algorithms. The primary comparison is based on the CEC2013 benchmarks, and we also perform several comparisons on the older CEC2005 benchmarks as well as a widely used set of classical functions. We also evaluate the effect of the size of the historical memory used by SHADE. Section VII concludes the paper with a discussion and directions for future work.

II. Differential Evolution

This section briefly describes DE [1]. 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 in population are initialized randomly. Then, a process of trial vector generation and selection are repeated until some termination criterion is encountered. In each generation G, a mutant vector $\boldsymbol{v}_{i,G}$ is generated from an existing population member $\boldsymbol{x}_{i,G}$ by applying some mutation strategy. The following are example of mutation strategies.

- rand/1

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{r_1,G} + F \cdot (\boldsymbol{x}_{r_2,G} - \boldsymbol{x}_{r_3,G}) \quad (1)$$

- rand/2

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{r_1,G} + F \cdot (\boldsymbol{x}_{r_2,G} - \boldsymbol{x}_{r_3,G}) + F \cdot (\boldsymbol{x}_{r_4,G} - \boldsymbol{x}_{r_5,G}) \quad (2)$$

- best/1

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{best,G} + F \cdot (\boldsymbol{x}_{r_1,G} - \boldsymbol{x}_{r_2,G}) \quad (3)$$

978-1-4799-0454-9/13/$31.00 ©2013 IEEE

- current-to-best/1

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + F \cdot (\boldsymbol{x}_{best,G} - \boldsymbol{x}_{i,G}) + F \cdot (\boldsymbol{x}_{r_1,G} - \boldsymbol{x}_{r_2,G}) \quad (4)$$

The indices $r_1, ..., r_5$ are randomly selected from $[1, N]$ such that they differ from each other as well as i. $\boldsymbol{x}_{best,G}$ is the best individual in population in generation G. The parameter $F \in [0, 1]$ controls the magnitude of the differential mutation operator.

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 $\boldsymbol{u}_{i,G}$. Binomial Crossover, the most commonly used crossover operator in DE, is implemented as follows:

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

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

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.

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

III. Related Work

The search efficiency of DE depends largely on the mutation strategy, as well as the control parameters N, F, CR, and many previous researchers have shown that optimal settings for these parameters are domain-dependent (c.f. [1], [8], [9], [10]). Thus, there has been much recent work which seeks to automate the selection of the mutation strategy, as well as the control parameter values [3], [2].

In this section, we briefly review several, state-of-the-art variants of DE which have been shown to perform relatively well. In our descriptions below, we say that a generation of trial vector is *successful* if a replacement occurs in Eq. (6), i.e., the trial vector $\boldsymbol{u}_{i,G}$ is more fit than the parent $\boldsymbol{x}_{i,G}$. Otherwise, we say that it is a *failure*.

jDE [11] assigns a different set of parameter values F_i and CR_i to each $\boldsymbol{x}_i$, which is used for generating the trial vectors. Initially, the parameters for all individuals i are set to $F_i = 0.5$, $CR_i = 0.9$. The control parameter values of the trial vectors are inherited from their parents. However, each parameter is randomly modified (within a pre-specified range) with some probability, and modified parameters are kept for the next generation only when a trial is successful. An extension of jDE which periodically reduces the population size by half in order to focus the search has been proposed [7].

SaDE [12] uses a memory of past search behavior in order to adapt the mutation strategy and control parameter values. First, 4 strategies are stored in a strategy pool, and during the search, in each generation, one of these strategies is selected probabilistically for each individual.

For each mutation strategy k $(k = 1, ..., K)$, the number of successes and failures is recorded. The randomized strategy selection is biased to favor strategies with a higher number of success (i.e., strategies that are adapted to the given problem). In each generation, the value of F for each individual $\boldsymbol{x}_i$ is randomly assigned by the normal distribution $\mathrm{randn}(0.5, 0.3)$, and does not adapt during the search. In contrast, successful CR values for each mutation strategy k are stored in corresponding memory k, and new values for CR are generated by sampling a normal distribution $\mathrm{randn}(CRm_k, 0.1)$, where CRm_k is the median CR value in memory for strategy k.

Although the memory mechanism of SaDE is similar to the historical memory mechanism used in SHADE, we do not experimentally compare SHADE to SaDE because we compare SHADE to CoDE, EPSDE, and JADE, which have already been shown to outperform SaDE [4], [6], [5].

EPSDE [6] uses 3 separate pools, one each for the mutation strategy, F, and CR. The mutation strategy pool includes rand/1/bin, best/2/bin, current-to-rand/1. The F pool stores the values between $[0.4, 0.9]$ in 0.1 increments, and the CR pool includes the values in $[0.1, 0.9]$ in 0.1 increments. At the beginning of the search, each individual $\boldsymbol{x}_{i,G}$ is randomly assigned values for the mutation strategy, F, and CR from each pool. During search, successful parameter sets are inherited by the individual in the next generation, $\boldsymbol{x}_{i,G+1}$. Parameter sets that fail are reinitialized.

Unlike the other self-adaptive DEs described in this section, CoDE [5], another recent DE variant, does not adapt its strategy and parameter settings, but randomly selects parameters from a predefined set. More specifically, at each generation, CoDE randomly combines 3 hand-selected mutation strategies (rand/1/bin, rand/2/bin, current-to-rand/1) with 3 hand-selected $[F, CR]$ pairs ($[1.0, 0.1]$, $[1.0, 0.9]$, $[0.8, 0.2]$), when generating 3 trial vectors for each individual in the population.

IV. Review of JADE [4]

In this section, we describe JADE [4], which is the basis for our SHADE algorithm. Its main features are a new mutation strategy (current-to-*p*best/1), an external archive, and adaptive control of the F, CR parameter values. Below, we describe each of these in detail and survey previous extensions to JADE.

A. current-to-pbest/1 mutation strategy

Since the current-to-best/1 strategy Eq. (4) directs the generation of mutant vectors towards the best member of the population, this greedy strategy converges quickly to a local optimum, and performs well on unimodal optimization problems. However, it has been shown that on multimodal problems, this strategy often leads to premature convergence and poor performance [10].

The mutation strategy used by JADE current-to-*p*best/1 is a variant of the current-to-best/1 strategy where the greediness is adjustable using a parameter p.

- current-to-*p*best/1

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + F_i \cdot (\boldsymbol{x}_{pbest,G} - \boldsymbol{x}_{i,G}) + F_i \cdot (\boldsymbol{x}_{r1,G} - \boldsymbol{x}_{r2,G}) \quad (7)$$

In Eq. (7), the individual $\boldsymbol{x}_{pbest,G}$ is randomly selected from the top $N \times p$ ($p \in [0, 1]$) members in the G-th generation. F_i is the F parameter used by individual $\boldsymbol{x}_i$. The greediness of current-to-$p$best/1 depends on the control parameter p in order to balance exploitation and exploration (small p behaves more greedily).

B. External Archive

In order to to maintain diversity, JADE uses an optional, external archive. Parent vectors $\boldsymbol{x}_{i,G}$ which were worse than the trial vectors $\boldsymbol{u}_{i,G}$ (and are therefore not selected for survival in the standard DE, Eq. 6) are preserved. When the archive is used, $\boldsymbol{x}_{r2,G}$ in Eq. (7) is selected from $\boldsymbol{P} \cup \boldsymbol{A}$, the union of the population $\boldsymbol{P}$ and the archive $\boldsymbol{A}$. The size of the archive is set to the same as that of the population, i.e., $|\boldsymbol{A}| = |\boldsymbol{P}|$. Whenever the size of the archive exceeds $|\boldsymbol{A}|$, randomly selected elements are deleted to make space for the newly inserted elements.

C. Parameter Adaptation

Each individual $\boldsymbol{x}_i$ is associated with its own CR_i and F_i parameters and generates trial vectors according to these values. These parameters are set probabilistically at the beginning of each generation according to adaptive control parameters μ_{CR}, μ_F according to the following equations:

$$CR_i = \mathrm{randn}_i(\mu_{CR}, 0.1) \tag{8}$$

$$F_i = \mathrm{randc}_i(\mu_F, 0.1) \tag{9}$$

Here, $\mathrm{randn}_i(\mu, \sigma^2)$, $\mathrm{randc}_i(\mu, \sigma^2)$ are values selected randomly from normal and Cauchy distributions with mean μ and variance σ^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. (9) is repeatedly applied to try to generate a valid value. At the beginning of the search, μ_{CR} and μ_F are both initialized to 0.5, and adapted during the search as follows.

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, μ_{CR}, μ_F are updated as:

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot \mathrm{mean}_A(S_{CR}) \tag{10}$$

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot \mathrm{mean}_L(S_F) \tag{11}$$

Here, the meta-level control parameter c is a learning rate (Zhang and Sanderson suggest a value of $c = 0.1$). $\mathrm{mean}_A(\cdot)$ is an arithmetic mean, and $\mathrm{mean}_L(\cdot)$ is a Lehmer mean which is is computed as:

$$\mathrm{mean}_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \tag{12}$$

D. Previous Extensions to JADE

Because JADE's update formula for μ_{CR} Eq. (10) uses an arithmetic mean, this biases μ_{CR} to converge to a small value. To prevent this bias, Peng et al proposed replacing the second term of Eq. (10) with the following weighted mean [13] :

$$\mathrm{mean}_{WA}(S_{CR}) = \sum_{k=1}^{|S_{CR}|} w_k \cdot S_{CR,k} \tag{13}$$

$$w_k = \frac{\Delta f_k}{\sum_{k=1}^{|S_{CR}|} \Delta f_k} \tag{14}$$

where $\Delta f_k = |f(\boldsymbol{u}_{k,G}) - f(\boldsymbol{x}_{k,G})|$. In the same paper, they also proposed a restart strategy for JADE [13].

Gong et al proposed an approach which adaptively selects among combines four configurations of JADE: the current-to-$p$best/1 mutation strategy with/without the external archive, and rand-to-p-best/1 mutation strategy [14] with/without the external archive [15].

In order to solve very high dimensional ($D = 1000$) problems, Yang et al propose a co-evolutionary extension to JADE [16]. JADE has also been adapted and applied to combinatorial optimization problems, as well as multi-objective optimization [17], [18].

V. Success-History based Adaptive DE

As described above, in each generation, JADE continuously updates μ_{CR}, μ_F such that they approach S_{CR}, S_F, which are the mean values for CR and F that have been successful in previous generations. While it is implicitly assumed that S_{CR} and S_F only includes parameter values which perform well on the given problem, due to the probabilistic nature of DE, it is possible that poor settings for CR and F are also included in S_{CR} and S_F. If so, Equations (10) and (11) can cause μ_{CR}, μ_F to move towards undesirable values, resulting in degraded search performance.

In order to improve upon the robustness of JADE, we propose Success-History based Adaptive DE (SHADE), an improved version of JADE which uses a different parameter adaptation mechanism based on a historical record of successful parameter settings. In SHADE, the mean values of S_{CR}, S_F for each generation are stored in a historical memory M_{CR}, M_F. In contrast to JADE, which uses a single pair (μ_{CR}, μ_F) to guide parameter adaptation, SHADE maintains a diverse set of parameters to guide control parameter adaptation as search progresses. Thus, even if S_{CR}, S_F for some particular generation contains a poor set of values, the parameters stored in memory from previous generations can not be directly, negatively impacted. This should result in SHADE being more robust than JADE.

In this section, we first describe the history based parameter adaptation strategy used by SHADE (Section V-A). We then describe a randomized approach to setting the control parameter value p in the current-to-$p$best/1 mutation strategy used by SHADE in Section V-B. Section V-C describes the overall SHADE algorithm, and Section V-D compares SHADE to previous work.

A. History Based Parameter Adaptation

As shown in Figure 1, SHADE maintains a historical memory with H entries for both of the DE control parameters

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

CR and F, M_{CR}, M_F. In the beginning, the contents of $M_{CR,i}, M_{F,i} (i = 1, ..., H)$ are all initialized to 0.5.

In each generation, the control parameters CR_i and F_i used by each individual $\boldsymbol{x}_i$ are generated by first selecting an index r_i randomly from $[1, H]$, and then applying the equations below:

$$CR_i = \text{randn}_i(M_{CR,r_i}, 0.1) \tag{15}$$

$$F_i = \text{randc}_i(M_{F,r_i}, 0.1) \tag{16}$$

If the values generated for CR_i, F_i are outside the range $[0, 1]$, they are adjusted/regenerated according to the procedure described above for JADE (Section IV-C).

As with JADE, the CR_i and F_i values used by successful individuals are recorded in S_{CR} and S_F, and at the end of the generation, the contents of memory are updated as follows:

$$M_{CR,k,G+1} = \begin{cases} \text{mean}_{WA}(S_{CR}) & \text{if } S_{CR} \neq \emptyset \\ M_{CR,k,G} & \text{otherwise} \end{cases} \tag{17}$$

$$M_{F,k,G+1} = \begin{cases} \text{mean}_{WL}(S_F) & \text{if } S_F \neq \emptyset \\ M_{F,k,G} & \text{otherwise} \end{cases} \tag{18}$$

An index k $(1 \leq k \leq H)$ determines the position in the memory to update. 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 generation G, the k-th element in the memory is updated. In the update equations (17) and (18), 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.

Also, the weighted mean $\text{mean}_{WA}(S_{CR})$ is computed according to Equation (13) by Peng et al [13]. The weighted Lehmer mean $\text{mean}_{WL}(S_F)$ is computed using the formula below, and as with $\text{mean}_{WA}(S_{CR})$, the amount of improvement is used in order to influence the parameter adaptation.

$$\text{mean}_{WL}(S_F) = \frac{\sum_{k=1}^{|S_F|} w_k \cdot S_{F,k}^2}{\sum_{k=1}^{|S_F|} w_k \cdot S_{F,k}} \tag{19}$$

B. Random generation of p

Although JADE adapts the DE control parameters CR and F, the parameter p, which is used to adjust the greediness of the current-to-$p$best/1 mutation strategy, is static and set manually. In SHADE, each individual $\boldsymbol{x}_i$ has an associated p_i, which is set according to the equation by generation:

$$p_i = \text{rand}[p_{min}, 0.2] \tag{20}$$

where p_{min} is set such that when the $p$best individual [see IV-A] is selected, at least 2 individuals are selected, i.e., $p_{min} = 2/N$. The maximum value 0.2 in Eq. (20) is the maximum value of the range for p suggested by Zhang and Sanderson [4].

Algorithm 1: SHADE

```
   //  Initialization phase
1  G = 0;
2  Initialize population P_0 = (x_{1,0}, ..., x_{N,0}) randomly;
3  Set all values in M_CR, M_F to 0.5;
4  Archive A = ∅;
5  Index counter k = 1;
   //  Main loop
6  while The termination criteria are not met do
7      S_CR = ∅, S_F = ∅;
8      for i = 1 to N do
9          r_i = Select from [1, H] randomly;
10         CR_{i,G} = randn_i(M_{CR,r_i}, 0.1);
11         F_{i,G} = randc_i(M_{F,r_i}, 0.1);
12         p_{i,G} = rand[p_min, 0.2];
13         Generate trial vector u_{i,G} by current-to-pbest/1/bin;
14     end
15     for i = 1 to N do
16         if f(u_{i,G}) ≤ f(x_{i,G}) then
17             x_{i,G+1} = u_{i,G};
18         else
19             x_{i,G+1} = x_{i,G};
20         end
21         if f(u_{i,G}) < f(x_{i,G}) then
22             x_{i,G} → A;
23             CR_{i,G} → S_CR, F_{i,G} → S_F;
24         end
25     end
26     Whenever the size of the archive exceeds |A|, randomly
       selected individuals are deleted so that |A| ≤ |P|;
27     if S_CR ≠ ∅ and S_F ≠ ∅ then
28         Update M_{CR,k}, M_{F,k} based on S_CR, S_F;
29         k++;
30         If k > H, k is set to 1;
31     end
32 end
```

C. Overall implementation

Algorithm 1 shows the overall SHADE algorithm, including the use of the external archive (as with JADE, the external archive is optional). In each generation, the DE control parameters F and CR are generated based on the history based parameter adaptation (Section V-A), trial vectors are generated, selection applied, and the historical memory (M_{CR}, M_F) is updated. This process is repeated until some termination criterion is achieved.

Algorithm 1 is similar to the JADE algorithm. However, note that in lines 21-24, the CR_i and F_i parameters are only updated when $f(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G})$. This constraint is necessary because the update equations are based on weighted score differences, so when $f(\boldsymbol{u}_{i,G})$ and $f(\boldsymbol{x}_{i,G})$ are identical, the weight becomes 0, resulting in inappropriate parameter updates.

D. Relationship With Previous Work

This section clarifies the relationship between the historical memory used by SHADE algorithm and related elements in previous work. While JADE has a learning rate parameter c which controls the rate of parameter adaptation, SHADE does not have an explicit learning rate parameter – instead, the memory size H plays a similar role. If H is small, then recent parameter values are frequently used (because older values are rapidly overwritten due to the limited memory size), leading to rapid convergence of the control parameter

TABLE I: Comparison of SHADE with state-of-the-art DE algorithms on the CEC2013 benchmarks. For all problems, the dimensionality $D = 30$, and the maximum number of objective function evaluations is $D \times 10,000 = 300,000$. All results are the means of 51 runs.

F	SHADE Mean (Std Dev)	CoDE Mean (Std Dev)	EPSDE Mean (Std Dev)	JADE Mean (Std Dev)	dynNP-jDE Mean (Std Dev)
F_1	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
F_2	9.00e+03 (7.47e+03)	9.78e+04 (4.81e+04)−	1.37e+06 (5.23e+06)−	**7.67e+03 (5.66e+03)**≈	9.52e+04 (4.09e+04)−
F_3	**4.02e+01 (2.13e+02)**	1.08e+06 (3.03e+06)−	1.75e+08 (5.39e+08)−	4.71e+05 (2.35e+06)−	1.71e+06 (2.54e+06)−
F_4	**1.92e-04 (3.01e-04)**	8.18e-02 (1.09e-01)−	8.08e+03 (2.56e+04)−	6.09e+03 (1.33e+04)−	4.76e+01 (4.75e+01)−
F_5	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
F_6	**5.96e-01 (3.73e+00)**	4.16e+00 (9.00e+00)−	9.27e+00 (1.33e+00)−	2.07e+00 (7.17e+00)≈	1.19e+01 (1.66e+00)−
F_7	4.60e+00 (5.39e+00)	9.32e+00 (6.34e+00)−	5.88e+01 (4.29e+01)−	3.16e+00 (4.13e+00)≈	**2.62e+00 (1.59e+00)**≈
F_8	**2.07e+01 (1.76e-01)**	2.08e+01 (1.18e-01)≈	2.09e+01 (5.32e-02)−	2.09e+01 (4.93e-02)−	2.10e+01 (3.98e-02)−
F_9	2.75e+01 (1.77e+00)	**1.45e+01 (2.90e+00)**+	3.50e+01 (4.21e+00)−	2.65e+01 (1.96e+00)+	2.20e+01 (5.12e+00)+
F_{10}	7.69e-02 (3.58e-02)	**2.71e-02 (1.50e-02)**+	1.02e-01 (5.65e-02)−	4.04e-02 (2.37e-02)+	3.63e-02 (2.34e-02)+
F_{11}	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	1.95e-02 (1.39e-01)≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
F_{12}	2.30e+01 (3.73e+00)	3.98e+01 (1.21e+01)−	4.94e+01 (9.28e+00)−	**2.29e+01 (5.45e+00)**≈	4.07e+01 (8.81e+00)−
F_{13}	5.03e+01 (1.34e+01)	8.04e+01 (2.74e+01)−	7.68e+01 (1.72e+01)−	**4.67e+01 (1.37e+01)**≈	7.10e+01 (1.72e+01)−
F_{14}	3.18e-02 (2.33e-02)	3.60e+00 (4.09e+00)−	3.99e-01 (6.00e-01)−	2.86e-02 (2.53e-02)≈	**9.39e-03 (1.40e-02)**+
F_{15}	**3.22e+03 (2.64e+02)**	3.36e+03 (5.31e+02)−	6.75e+03 (7.60e+02)−	3.24e+03 (3.17e+02)≈	4.39e+03 (4.72e+02)−
F_{16}	9.13e-01 (1.85e-01)	**3.38e-01 (2.03e-01)**+	2.48e+00 (2.88e-01)−	1.84e+00 (6.27e-01)−	2.32e+00 (2.83e-01)−
F_{17}	**3.04e+01 (3.83e-14)**	3.04e+01 (1.17e-02)−	3.04e+01 (2.51e-02)−	3.04e+01 (1.95e-14)−	3.04e+01 (1.78e-03)≈
F_{18}	7.25e+01 (5.58e+00)	**6.69e+01 (9.23e+00)**+	1.37e+02 (1.12e+01)−	7.76e+01 (5.91e+00)−	1.35e+02 (1.24e+01)−
F_{19}	1.36e+00 (1.20e-01)	1.61e+00 (3.58e-01)−	1.84e+00 (2.00e-01)−	1.44e+00 (8.71e-02)−	**1.27e+00 (1.09e-01)**+
F_{20}	1.05e+01 (6.04e-01)	1.06e+01 (6.69e-01)≈	1.30e+01 (6.33e-01)−	**1.04e+01 (5.82e-01)**≈	1.13e+01 (4.14e-01)−
F_{21}	3.09e+02 (5.65e+01)	3.02e+02 (9.02e+01)≈	3.05e+02 (8.06e+01)≈	3.04e+02 (6.68e+01)≈	**2.94e+02 (8.29e+01)**≈
F_{22}	9.81e+01 (2.52e+01)	1.17e+02 (9.96e+00)−	3.09e+02 (1.12e+02)−	**9.39e+01 (3.08e+01)**≈	1.03e+02 (2.57e+01)−
F_{23}	3.51e+03 (4.11e+02)	3.56e+03 (6.12e+02)≈	6.74e+03 (8.20e+02)−	**3.36e+03 (4.01e+02)**≈	4.36e+03 (4.61e+02)−
F_{24}	2.05e+02 (5.29e+00)	2.21e+02 (9.28e+00)−	2.91e+02 (7.08e+00)−	2.17e+02 (1.57e+01)−	**2.04e+02 (4.22e+00)**≈
F_{25}	2.59e+02 (1.96e+01)	2.57e+02 (6.55e+00)≈	2.99e+02 (3.29e+00)−	2.74e+02 (1.06e+01)−	**2.55e+02 (7.91e+00)**≈
F_{26}	2.02e+02 (1.48e+01)	2.18e+02 (4.48e+01)−	3.56e+02 (6.49e+01)−	2.15e+02 (4.11e+01)≈	**2.00e+02 (3.06e-03)**+
F_{27}	**3.88e+02 (1.09e+02)**	6.20e+02 (1.01e+02)−	1.21e+03 (7.42e+01)−	6.70e+02 (2.40e+02)−	3.90e+02 (9.12e+01)≈
F_{28}	**3.00e+02 (0.00e+00)**	**3.00e+02 (0.00e+00)**≈	**3.00e+02 (0.00e+00)**≈	**3.00e+02 (0.00e+00)**≈	**3.00e+02 (0.00e+00)**≈
	+	4	0	2	5
	-	15	23	10	13
	≈	9	5	16	10

values. On the other hand, as H increases, the rate of control parameter convergence is expected to slow down because older parameters will continue to have influence for a longer time.

The use of historical memory in SHADE is similar to that of SaDE (Section III), which uses a similar memory mechanism in order to adapt the CR parameter and the mutation strategy selection [12]. In SaDE, CR_i, the CR value for the ith individual is randomly generated from a normal distribution $\mathrm{randn}_i(CRm_k, 0.1)$, whose mean is the median CR value in memory for strategy k. In contrast, in SHADE, (1) the actual parameter values for CR, and F that were used by individuals are not directly stored in memory as in SaDE – instead, only the means of the successful parameter sets S_{CR}, S_F for each generation are stored, and (2) rather than generating new control parameter values based on a random distribution around the median memory elements as SaDE does, SHADE randomly selects a particular element from memory, and generates a new parameter value from a distribution based around that particular element.

VI. Experimental Results

SHADE was evaluated on the CEC2013 benchmark problem set [19], as well as CEC2005 benchmarks [20] and the set of 13 "classical" benchmark problems [21] that have been frequently used in the literature. We compared SHADE to the following, state-of-the-art DE algorithms:

- JADE [4],
- dynNP-jDE [7] (an improved version of jDE [11]),
- CoDE [5], and
- EPSDE [6]

For each algorithm, we used the control parameter values that were suggested in the cited, original papers. SHADE used a population size $N = 100$ and memory size $H = N = 100$ for this experiment (unless explicitly noted, all SHADE runs in this paper used these parameters).

The Matlab source code for CoDE, EPSDE, and JADE were downloaded from [22], the web site of Q. Zhang's, one of the authors of [5]. These were used for the experiments in [5], and are based on code originally received from the original

authors of CoDE, EPSDE, and JADE. We minimally modified these programs so that they would work with the CEC2013 benchmark codes.

SHADE and dynNP-jDE were implemented in C++. In addition, for each dimension j, if the mutant vector element $v_{j,i,G}$ is outside the boundaries $[x_j^{min}, x_j^{max}]$, we applied the same correction performed in [4]:

$$v_{j,i,G} = \begin{cases} (x_j^{min} + x_{j,i,G})/2 & \text{if } v_{j,i,G} < x_j^{min} \\ (x_j^{max} + x_{j,i,G})/2 & \text{if } v_{j,i,G} > x_j^{max} \end{cases} \quad (21)$$

A. Results on the CEC2013 benchmarks

We evaluated SHADE and the other DE variants on the 28 benchmark problems from the *CEC2013 Special Session on Real-Parameter Single Objective Optimization* benchmark suite. [19]. We use the CEC2013 benchmark problems because of a precision-related issue with earlier benchmarks (CEC2005) which arose when comparing our C++ code for SHADE and dynNP-jDE and the publicly available Matlab code for CoDE, EPSDE and JADE that we downloaded from [22]. Since the precisions achievable with these codes were different, a completely fair comparison based on the complete set of CEC2005 benchmarks was difficult (however, see below in Section VI-B for comparisons on problems where precision-related issues did not arise). Fortunately, the CEC2013 benchmarks and rules were designed to overcome this issue.

Functions $F_1 \sim F_5$ are unimodal, and , $F_6 \sim F_{20}$ are multimodal. $F_{21} \sim F_{28}$ are composite functions which combine multiple test problems into a complex landscape. See [19] for details.

We performed our evaluation following the guidelines of the CEC2013 benchmark competition [19]. For all of the problems, the search space is $[-100, 100]^D$. 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 $D = 30$, and the maximum number of objective function calls per run was $D \times 10,000$ (i.e., 300,000). The number of runs per problem was 51, and the average performance of these runs was evaluated. Statistical significance testing was done using the Wilcoxon rank-sum test with (significant threshold: $p < 0.05$).

The results are shown in Table I. In the table, the mean and standard deviation of the error (difference) between the best fitness values found in each run and optimal value are shown. The best result for each problem is shown in boldface. The $+, -, \approx$ indicate whether a given algorithm performed significantly better ($+$), significantly worse ($-$), or not significantly different better or worse ($\approx$) compared to SHADE according to the Wilcoxon rank-sum test (significance threshold $p < 0.05$).

On the unimodal problems $F_1 \sim F_5$ SHADE and JADE exhibited the best performance. The good performance of JADE on the unimodal functions is consistent with the previous results of [5]. On the basic multimodal functions $F_6 \sim F_{20}$, SHADE performs relatively well, although there are several problems where CoDE performed particularly well. On the complex, composite functions $F_{21} \sim F_{28}$, the best performers were dynNP-jDE (possibly due the its population size reduction strategy), followed by SHADE, JADE and CoDE. Finally, as shown in the bottom three rows of the table counting the number of $+$,$-$, and $\approx$ results, SHADE has the best overall performance on these 28 problems.

TABLE II: Comparison of SHADE with CoDE on the CEC2005 benchmark functions (30 dimensions).

F	SHADE Mean (Std)	CoDE Mean (Std)
F_1	8.05e-18 (1.27e-17)	**0.00e+00 (0.00e+00)**+
F_2	**2.53e-17 (9.74e-18)**	2.29e-15 (5.26e-15)−
F_3	**1.18e+04 (8.43e+03)**	1.06e+05 (5.63e+04)−
F_4	**8.45e-10 (7.03e-09)**	4.40e-03 (1.19e-02)−
F_5	**1.18e-03 (8.87e-03)**	4.35e+02 (4.39e+02)−
F_6	2.79e-01 (1.02e+00)	**1.59e-01 (7.85e-01)**+
F_7	9.88e-03 (6.98e-03)	**8.74e-03 (9.41e-03)**+
F_8	2.03e+01 (3.69e-01)	**2.01e+01 (1.44e-01)**≈
F_9	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈
F_{10}	**2.36e+01 (3.47e+00)**	4.07e+01 (1.15e+01)−
F_{11}	2.66e+01 (1.93e+00)	**1.20e+01 (3.32e+00)**+
F_{12}	**1.60e+03 (2.34e+03)**	3.15e+03 (4.26e+03)−
F_{13}	**1.36e+00 (9.64e-02)**	1.58e+00 (3.13e-01)−
F_{14}	1.24e+01 (2.55e-01)	**1.23e+01 (4.54e-01)**≈
F_{15}	**3.58e+02 (9.34e+01)**	4.00e+02 (6.96e+01)−
F_{16}	7.40e+01 (8.82e+01)	**7.11e+01 (3.96e+01)**+
F_{17}	9.20e+01 (6.89e+01)	**7.04e+01 (2.73e+01)**+
F_{18}	**9.02e+02 (1.82e+01)**	9.05e+02 (1.03e+00)−
F_{19}	**9.05e+02 (1.07e+01)**	9.05e+02 (1.01e+00)−
F_{20}	**9.04e+02 (1.07e+01)**	9.04e+02 (8.43e-01)−
F_{21}	5.03e+02 (3.00e+01)	**5.00e+02 (0.00e+00)**≈
F_{22}	8.78e+02 (1.33e+01)	**8.60e+02 (2.16e+01)**+
F_{23}	5.38e+02 (4.03e+01)	**5.34e+02 (4.28e-04)**+
F_{24}	**2.00e+02 (0.00e+00)**	**2.00e+02 (0.00e+00)**≈
F_{25}	**2.09e+02 (1.46e-01)**	2.11e+02 (9.29e-01)−
	+	8
	-	12
	≈	5

B. Evaluation of SHADE in Previously Published Environments

In the previous section, we showed that SHADE outperformed previous DE variants on the CEC2013 benchmarks. However, we did not specifically tune the other algorithms for the CEC2013 benchmarks, and used the control parameter values which were suggested by the authors of the original papers. Because the CEC2013 problems differ from the benchmarks used in the original papers which are the source of these suggested control parameter values, it is possible that these suggested values were not appropriate for the CEC2013 benchmarks, resulting in an unfair comparison that favored SHADE.

Therefore, in this section, we evaluated SHADE against the previous DE variants in environments that are as close as possible to the environments described in the original papers. This set of comparisons does not include EPSDE (which is included in the CEC2013 comparisons above), because the original paper on EPSDE [6] uses a set of original benchmark

TABLE III: Comparison of SHADE with JADE on the 13 classical functions (30 dimensions).

f	Gen	SHADE Mean (Std Dev)	JADE Mean (Std Dev)
f_1	1500	**1.0e-70 (4.4e-70)**	1.3e-54 (9.2e-54)
f_2	2000	**4.5e-49 (5.1e-49)**	3.9e-22 (2.7e-21)
f_3	5000	5.4e-64 (3.3e-63)	**6.0e-87 (1.9e-86)**
f_4	5000	2.4e-41 (9.6e-41)	**4.3e-66 (1.2e-65)**
f_5	3000	**8.0e-02 (5.6e-01)**	3.2e-01 (1.1e+00)
	20000	**8.0e-02 (5.6e-01)**	3.2e-01 (1.1e+00)
f_6	100	**2.7e+00 (1.2e+00)**	5.6e+00 (1.6e+00)
	1500	**0.0e+00 (0.0e+00)**	**0.0e+00 (0.0e+00)**
f_7	3000	**5.8e-04 (2.2e-04)**	6.8e-04 (2.5e-04)
f_8	1000	**1.4e-03 (1.7e-03)**	7.1e+00 (2.8e+01)
	9000	**0.0e+00 (0.0e+00)**	7.1e+00 (2.8e+01)
f_9	1000	1.6e-02 (7.4e-03)	**1.4e-04 (6.5e-05)**
	5000	**0.0e+00 (0.0e+00)**	**0.0e+00 (0.0e+00)**
f_{10}	500	**2.5e-10 (9.4e-11)**	3.0e-09 (2.2e-09)
	2000	5.5e-15 (1.8e-15)	**4.4e-15 (0.0e+00)**
f_{11}	500	**1.5e-14 (9.3e-14)**	2.0e-04 (1.4e-03)
	3000	**0.0e+00 (0.0e+00)**	2.0e-04 (1.4e-03)
f_{12}	500	**3.7e-19 (1.2e-18)**	3.8e-16 (8.3e-16)
	1500	**1.6e-32 (0.0e+00)**	**1.6e-32 (5.5e-48)**
f_{13}	500	**3.9e-18 (5.6e-18)**	1.2e-15 (2.8e-15)
	1500	**1.3e-32 (0.0e+00)**	1.4e-32 (1.1e-47)

problems which were nontrivial to replicate faithfully. All comparisons below use 30-dimensional problems.

We compared SHADE to CoDE using the CEC2005 benchmark [20]. Initially, we compared the SHADE results on the CEC2005 benchmarks to the CEC2005 benchmark results for CoDE in Table I of [5]. However, because the distributions of the data in [5] overlap with the SHADE results, it was unclear whether the differences between the two algorithms was significant. Therefore, in order to be able to perform statistical significance testing to determine which algorithm performed better, we executed the CoDE Matlab code obtained from [22].

We compared SHADE to the original published results JADE [4], and dynNP-jDE [7] by running SHADE on the same problems and under the same experimental conditions described in the original papers. Both the comparisons with JADE and dynNP-jDE were performed on the 13 "Classical functions" [21]. The JADE results in Table III are from Table IV in [4], and the dynNP-jDE results in Table IV are from Table 3 in [7].

Tables II, III and IV show the results of the comparisons with CoDE, JADE, and dynNP-jDE as described above. The maximum number of generations and objective function calls per run in JADE and dynNP-jDE are shown in Tables III and IV. For CoDE, the maximum number of objective function calls per run was $D \times 10,000$ (i.e., 300,000). The number of runs per problem in CoDE, JADE and dynNP-JDE was 100, 50 and 100 respectively. It can be seen that in each case, SHADE performs better.

However, note that some performance differences are most likely not due to fundamental algorithmic differences, but rather due to the implementation language and experimental environment. For example, in Table III, on problem f_{13}, the scores achieved by SHADE on the final generation appear to be better than the scores achieved by JADE. However this is actually likely to be due the difference in precision between our C++ code for SHADE and the publicly available code for JADE that we downloaded from [22].

TABLE IV: Comparison of SHADE with dynNP-jDE on the 13 classical functions (30 dimensions).

f	Max NFE	SHADE Mean (Std Dev)	dynNP-jDE Mean (Std Dev)
f_1	1.5e+05	**8.76e-71 (3.63e-70)**	4.04e-49 (3.76e-48)
f_2	2.0e+05	**3.78e-49 (4.40e-49)**	3.53e-44 (2.15e-43)
f_5	2.0e+06	1.20e-01 (6.83e-01)	**1.54e-28 (1.76e-28)**
f_6	1.5e+05	**0 (0)**	**0 (0)**
f_7	3.0e+05	**6.15e-04 (2.25e-04)**	1.69e-3 (4.92e-4)
f_8	9.0e+05	**-12569.5 (0.00e+00)**	**-12569.5 (4.73e-11)**
f_9	5.0e+05	**0 (0)**	**0 (0)**
f_{10}	1.5e+05	5.29e-15 (1.78e-15)	**3.73e-15 (7.78e-16)**
f_{11}	2.0e+05	**0 (0)**	**0 (0)**
f_{12}	1.5e+05	**1.57e-32 (0.00e+00)**	1.92e-32 (3.69e-32)
f_{13}	1.5e+05	**1.35e-32 (0.00e+00)**	3.40e-32 (9.36e-32)

C. Impact of Memory size

SHADE has two parameters that need to be set by the user: the population size, and the memory size H. We investigated the impact of H on the search performance of SHADE.

Table V shows the performance of SHADE on the CEC2013 benchmarks, using various values for H. In the experiments reported above, H was set to 100. Here, we show results for $H \in \{5, 30, 50, 200, 300, 400, 500\}$. Due to space constraints, we do not show results for each benchmark function separately, but only show the aggregate results of statistical testing, i.e., for each value of H, the number of times it was significantly better/worse/equivalent to $H = 100$ according to the Wilcoxon rank-sum test ($p < 0.05$).

Relative to the results for $H = 100$, the results for $H = 30$ and $H = 50$ are slightly better. The results for $H = 5$ and $H = 10$ are approximately equivalent to the performance when $H = 100$. For $H > 100$, the performance seems to degrade monotonically as H increases. From these results, it can be concluded that the performance of SHADE depends on the memory size H, and that the value $H = 100$ seems to be a fairly good setting. A more complete understanding of the impact of H, especially on higher dimensional problems, is an avenue for future work.

D. On the importance of a finite memory

The previous experiment showed that if SHADE is implemented as shown in Algorithm 1, the performance is dependent on H, the size of the memory. Next, we investigated whether it might be possible to eliminate H altogether by evaluating variants of SHADE where $H = \infty$, and various mechanisms for selecting parameters from memory are used. In these

TABLE V: SHADE Results on CEC2013 benchmarks using various memory sizes H

H	5	10	30	50	100	200	300	400	500
	7	7	6	4	+	1	0	3	3
	6	7	2	1	-	6	8	10	12
	15	14	20	23	$\approx$	21	20	15	13

TABLE VI: Evaluation of infinite memory size SHADE variants (compared to $H = 100$)

H	100	∞-Random	∞-Time	∞-Fitness
	+	3	3	4
	-	14	8	18
	$\approx$	11	17	6

variants, the memory size is set to the maximum number of generations to be executed, which, in practice, is equivalent to $H = \infty$.

Furthermore, Algorithm 1, line 9 is modified so that the element that is selected from memory is selected in one of the following ways: (1) Randomly, (2) favor recently used parameters[1], (3) favor parameters that led to the largest improvements since the previous generation (a roulette-wheel selection). We refer to these variants as these variants as ∞-Random, ∞-Time, and ∞-Fitness, respectively.

The results are shown in Table VI. As with Section VI-C, we show only the aggregate results of the statistical comparisons of the infinite-memory SHADE variants compared to $H = 100$. All of the infinite-memory variants clearly performed worse than $H = 100$. Thus, it seems that the use of a properly tuned, finite memory size contributes significantly to the performance of SHADE.

VII. Conclusions

We proposed SHADE, an adaptive DE algorithm based which extends JADE [4] by using a historical memory of recently successful parameter sets in order to guide the generation of new control parameter values. SHADE was shown to outperform previous, state-of-the-art DE algorithms on a large set of benchmark problems, including the CEC2013 benchmark set, as well as CEC2005 benchmarks and the set of 13 classical benchmark problems.

Directions for future work include analysis of how the parameter values actually change during search, in order to gain a deeper understanding of the dynamics of the algorithm (e.g., how does the trajectory of the control parameters differ compared to strategies such as JADE, jDE, EPSDE), as well as developing a principled method for determining the memory size parameter for new classes of problems.

[1] In this case, P_i, the probability of selecting index G_i is determined using a roulette-wheel selection based on the value of G_i: $P_i = G_i / \sum_{j=1}^{CG} G_j$, where CG is number of the current generation.

References

[1] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, 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] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 61–106, 2010.

[4] 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.

[5] 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.

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

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

[8] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," in *Int. Conf. on Adv. in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, 2002, pp. 293–298.

[9] J. Rönkkönen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *IEEE CEC*, 2005, pp. 506–513.

[10] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *GECCO*, 2006, pp. 485–492.

[11] 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.

[12] 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.

[13] 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.

[14] J. Zhang and A. Sanderson, *Adaptive differential evolution: a robust approach to multimodal problem optimization*. Springer, 2009, vol. 1.

[15] W. Gong, Z. Cai, C. X. Ling, and H. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Trans. on Systems, Man, and Cybernetics, PartB*, vol. 41, no. 2, pp. 397–413, 2011.

[16] Z. Yang, J. Zhang, K. Tang, X. Yao, and A. C. Sanderson, "An adaptive coevolutionary differential evolution algorithm for large-scale optimization," in *IEEE CEC*, 2009, pp. 102–109.

[17] J. Zhang, V. Avasarala, A. C. Sanderson, and T. Mullen, "Differential evolution for discrete optimization: An experimental study on combinatorial auction problems," in *IEEE CEC*, 2008, pp. 2794–2800.

[18] J. Zhang and A. C. Sanderson, "Self-adaptive multi-objective differential evolution with direction information provided by archived inferior solutions," in *IEEE CEC*, 2008, pp. 2801–2810.

[19] 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.

[20] 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.

[21] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Tran. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, 1999.

[22] "http://dces.essex.ac.uk/staff/qzhang."