Revisiting Success-Histories for Adaptive Differential Evolution

Tomofumi Kitamura
Department of General Systems Studies
The University of Tokyo
Tokyo, Japan
kitamura-tomofumi094@g.ecc.u-tkyo.ac.jp

Alex Fukunaga
Department of General Systems Studies
The University of Tokyo
Tokyo, Japan
fukunaga@idea.c.u-tokyo.ac.jp

***Abstract*—The introduction of success-history based adaptation in SHADE, a variant of JADE, resulted in a significant advance in the performance of adaptive differential evolution. Many variants of SHADE which use the same success history mechanism have been proposed, but the success history mechanism has remained poorly understood. We revisit use of the success history based adaptation, and show experimentally that the standard approach to sampling from the success history in SHADE may not be as vital to performance as previously assumed. We show that EnJADE, a simple, new variant of JADE which maintains an ensemble of control parameter distribution means, can outperform SHADE on the CEC14 benchmark suite. We also show the effectiveness of the new ensemble-based approach when combined with linear population reduction.**

***Index Terms*—adaptive differential evolution, SHADE, JADE, ensemble methods**

I. Introduction

In continuous function optimization problems, the task is to find a real-valued vector $x = (x_1, \cdots, x_D)^T$ which minimizes the value of an objective function $f = \mathbb{R}^D \rightarrow \mathbb{R}$. In many real-world applications, the objective function f is provided as a "black box" (e.g,. simulation code) without access to its internal structure. Evolutionary algorithms are often applied to such continuous function optimization problems. Differential Evolution [1] is a widely used class of evolutionary algorithms where new individuals (candidate solution vectors for the function optimization problem) are generated based on difference vectors between previously generated individuals. DE has been applied to many real-world applications, and many variants of DE have been widely studied [2]–[4]. In recent years, there has been significant advances in *adaptive* DE methods which adapt their control parameters during runtime [5].

A widely used approach to control parameter adaptation chooses new parameter values based on parameter values which are deemed to have been useful earlier during the search. For example, JADE [6] stores the means of past, successful parameter values, and uses these means as the basis of a probability distribution from which new control parameter values are generated.

One particularly successful class of adaptive DE descended from JADE is based on SHADE [7]. The key difference between JADE and SHADE is that SHADE and its variants use a *success-history*, $MF = (MF_1, \cdots, MF_H)$ and $MC = (MC_1, \cdots, MC_H)$, a *per-generation* record of the means of scale factor (F) and crossover rate (C) parameter values which resulted in highly fit new vector generations,

Despite the success of SHADE and its many variants, the success history mechanism, which was the core contribution of SHADE, remains relatively poorly understood. In the original paper introducing SHADE, the primary intuitive motivation for success history mechanism was robustness: "In contrast to JADE, which uses a single pair (μ_C,μ_F) to guide parameter adaptation, SHADE maintains a diverse set of parameters to guide control parameter adaptation as search progresses. Thus, even if [the success histories] 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." [7](p.73,col 2, par 4). However, to our knowledge, the "robustness" aspect of the success history mechanism (in particular, the robustness vs. "poor" parameter values as described above) has not been empirically investigated.

Instead of robustness, most previous work on SHADE and its variants has focused on search performance, as it turned out that SHADE and its variants performed quite well on benchmark competitions (DEs which used the SHADE success history mechanism have been the top-ranked algorithms in the IEEE CEC single-objective benchmark optimization competitions from 2014-2017).

While theoretical analysis of adaptive DE mechanisms is difficult, it has been experimentally shown that compared to an oracle with access to optimal 1-step lookahead, the SHADE parameter adaptation mechanism generates (F, C) values closer to the values selected by a 1-step lookahead oracle than the JADE and jDE mechanisms [8]. However, the same study also showed that while the SHADE adaptation mechanism tracked the oracular values more closely than other mechanisms, there was still a large gap between SHADE and the oracle, i.e., there is still much room for improvement in DE parameter adaptation.

It is hypothesized in [8] that the desired, "correct" behavior of a DE parameter adaptation mechanism is to approximate the optimal (oracular) parameter values as closely as possible. However, if we consider the success-history mechanism with respect to this objective, the storage of per-generation means for many generations (typically $D/2$, where D is the dimensionality of the problem), is somewhat contradictory – it is not clear that generating the next pair of (F, C) based a random pair of (F, C) parameters which happened to be useful many generations ago results in better tracking of an ideal parameter sequence. This suggests that it may be useful to seek a different framework for interpreting the array of parameters used by SHADE's success history mechanism from a different perspective, possibly leading to a better intuitive understanding and perhaps allowing a better usage of the same data structure resulting in improved search performance.

This paper revisits the success history mechanism of SHADE. We first review the difference between the adaptation mechanisms of SHADE and its predecessor, JADE [6] (Sec. II. We evaluate alternate approaches to sampling the success history array in SHADE, and show always sampling from a fixed index in the history array can significantly outperform the standard approach of uniformly sampling from the history (Sec. III). We observe that this can be reinterpreted as a type of ensemble diversification mechanism built on top of the JADE adaptation mechanism, and propose Ensemble JADE (EnJADE), a variation of JADE which maintains an ensemble of independent parameter mean values (Sec. IV) among which search effort is allocated equally in a round-robin manner. We show experimentally that EnJADE is competitive with SHADE (Sec. V). Furthermore, we show that the ensemble strategy remains competitive when linear population reduction [9], a standard enhancement to SHADE, is used (Sec. VI). Finally, we consider robustness, the original motivation for the success history mechanism, and show experimentally a setting with noise injection, SHADE has considerably more stable search performance than JADE and EnJADE (Sec. VII).

II. Background

The basic DE algorithm proposed in [1] is as follows. A population $P = x_1, ..., x_N$ with N members, where each candidate individual $x_i \in P$ is a D-dimensional vector, is initialized randomly. At each iteration t, for each $x_{i,t}$, a mutant vector $v_{i,t}$ is generated. The mutant $v_{i,t}$ is recombined with its parent $x_{i,t}$ using some crossover operator to generate a *trial vector* $u_{i,t}$. After all trial vectors are generated, each trial vector $u_{i,t}$ is compared with its parent $x_{i,t}$, and the better of these two individuals is kept as the i-th individual in the next generation P_{t+1}. A trial vector $u_{i,t}$ is called *successful* if it has a better fitness value than its parent $x_{i,t}$.

Most of the widely used parameter adaptation mechanisms for DE adjust one or both of two parameters, F and C [5]. The Scale Factor F controls the magnitude of the mutation applied to to the parent $x_{i,t}$ to generate the mutant vector $v_{i,t}$. The Crossover Rate C controls the number of values in a new trial vector $u_{i,t}$ which are copied from its parent $x_{i,t}$.

There are many approaches for adapting these parameters. In one common approach, the generation of new (F, C) values is biased according to whether previous (F, C) values resulted in successful trial vectors. For example, JADE generates (F, C) values for the next iteration using a probability distribution based on the mean values of parameter values which have previously resulted in successful trial vectors [6]. cDE [10] maintains a pool of (F, C) parameter value pairs, and at each iteration, for each individual, it selects one of these pairs, where the probability of selection is based on their past successes. See [5] for an extensive survey of parameter adaptation mechanisms.

Algorithm 1 JADE

```
1:  $t \leftarrow 1$, Initialize population $P$;
2:  $\mu_F = 0.5, \mu_C = 0.5$;
3:  while not termination condition do
4:      $S_F \leftarrow \emptyset, S_C \leftarrow \emptyset$;
5:      for $i = 1$ to $N$ do
6:          $F_{i,t} = randc(\mu_F, 0.1)$;
7:          $C_{i,t} = randn(\mu_C, 0.1)$;
8:          generate $v_{i,t}$ using current-to-pbest mutation;
9:          generate $u_{i,t}$ using binomial crossover;
10:     for $i = 1$ to $N$ do
11:         if $f(u_{i,t}) \leq f(x_{i,t})$ then
12:             $x_{i,t+1} \leftarrow u_{i,t}$;
13:             $S_F \leftarrow S_F \cup F_{i,t}, S_C \leftarrow S_C \cup C_{i,t}$;
14:         else
15:             $x_{i,t+1} \leftarrow x_{i,t}$;
16:     if $S_F, S_C \neq \emptyset$ then
17:         $\mu_F \leftarrow (1-c) * \mu_F + c * mean_L(S_F)$;
18:         $\mu_C \leftarrow (1-c) * \mu_C + c * mean_L(S_C)$;
19:     $t \leftarrow t + 1$;
```

A. JADE

JADE [6], shown in Alg. 1, is an adaptive DE which had several distinguishing features that influenced many later adaptive DEs: (1) a new mutation strategy, the current-to-$p$best, (2) the use of an (optional) external archive which stored failed parent vectors in order to maintain a source of diversity, and (3) the control parameter adaptation strategy for F and C described below. Due to space, we review in detail only the parameter adaptation mechanism.

The parameters μ_F and μ_C are initialized to 0.5 at the beginning of the run (line 2). In each generation t, the sets S_F and S_C are initialized to the empty set (line 4). During trial vector generation (lines 5-9), for each x_i in the population, a trial vector is generated using current-to-pbest mutation and binomial crossover. For each such trial vector generation, a different $F_{i,t} = randc(\mu_F, 0.1)$ and $C_{i,t} = randn(\mu_C, 0.1)$ is used (lines 6-7), where $randc$ is the Cauchy distribution (randc is repeatedly called until value > 0 is generated, and then capped at 1), and $randn$ is the normal distribution. If the trial vector $u_{i,t}$ is successful (has a better fitness than its parent $x_{i,t}$), $F_{i,t}$ and $C_{i,t}$ are added to the sets S_F and S_C, respectively (line 13). At the end of a generation, after all of the trial vectors are generated and evaluated, the μ_F and μ_C parameters are updated as: $\mu_F \leftarrow (1-c)*\mu_F+c*mean_L(S_F)$

and $\mu_C \leftarrow (1-c) * \mu_C + c * mean_L(S_C)$, where $mean_L$ computes the Lehmer mean (lines 17-18).

The original JADE used the arithmetic mean instead of the Lehmer mean for updating μ_C in line 18.

B. SHADE

SHADE, shown in Alg. 2 [7] is based on JADE. The main difference is the parameter adaptation mechanisms. Instead of using a single pair of mean values μ_F and μ_C summarizing all past successful (F, C) parameters, SHADE *success history* data structure which stores the means of successful F and C values *per generation*, and uses this structure for (F, C) parameter generation.[1].

The success history for F, $MF = (MF_1, \cdots, MF_H)$, and the success history for C, $MC = (MC_1, \cdots, MC_H)$, are arrays of length H (H is usually set to $D/2$, where D is the dimensionality of the problem). Initially, all elements of MF and MC are initialized to 0.5 (line 2).

During search, the success histories are used and updated as follows. For each trial vector generation, an r_i, an index into the history vector is selected uniformly randomly from $[1, H]$ (line 7). Then, the $F_{i,t}$ and $C_{i,t}$, the F and C values used for the successor generation for $x_{i,t}$ are generated in lines 8-9 as:

$$F_{i,t} = min(randc(MF_r, 0.1), 1)$$

$$C_{i,t} = max(0, min(randn(MC_r, 0.1), 1))$$

C is generated using the normal distribution with mean MC and restricted to $[0, 1]$. F is generated using the Cauchy distribution with mean MF.

After all trial vectors are generated and evaluated, the success histories MF and MC are updated (lines 19-20) as follows. $MF_k = mean_L(S_F)$, and $MC_k = mean_L(S_C)$, where $mean_L(X)$ is the Lehmer mean of X, and $k \in \{1, ..., H\}$ is an index which is initialized to 1 at the beginning of search (line 3), and incremented (modulo H), whenever a new element is inserted into the history (line 21).

Comparing Algorithm 1 and 2, it can be seen that overall, JADE and its descendant, SHADE are quite similar, except for the use of the μ_F and μ_C means for $F_{i,t}$ and $C_{i,t}$ generation by JADE, and the use of the success history arrays MF and MC by SHADE. (In this paper, we focus on the parameter adaptation mechanisms, so we do not show archive-related details, which have some differences, in Algorithms 1-2). In this context, JADE can be seen as a special case of SHADE where H, the size of the success history array, is 1.

Algorithm 2 SHADE

```
1:  $t \leftarrow 1$, Initialize population $P$;
2:  Initialize contents of success histories $MF$ and $MC$ to 0.5;
3:  $k \leftarrow 1$;
4:  while not termination condition do
5:      $S_F \leftarrow \emptyset, S_C \leftarrow \emptyset$;
6:      for $i = 1$ to $N$ do
7:          select $r_i$ randomly from $\{1, \cdots, H\}$
8:          $F_{i,t} = randc(MF_{\mathbf{r_i}}, 0.1)$;
9:          $C_{i,t} = randn(MC_{\mathbf{r_i}}, 0.1)$;
10:         generate $v_{i,t}$ using current-to-pbest mutation;
11:         generate $u_{i,t}$ using binomial crossover;
12:     for $i = 1$ to $N$ do
13:         if $f(u_{i,t}) \leq f(x_{i,t})$ then
14:             $x_{i,t+1} \leftarrow u_{i,t}$;
15:             $S_F \leftarrow S_F \cup F_{i,t}, S_C \leftarrow S_C \cup C_{i,t}$;
16:         else
17:             $x_{i,t+1} \leftarrow x_{i,t}$;
18:     if $S_F, S_C \neq \emptyset$ then
19:         $MF_k \leftarrow mean_L(S_F)$;
20:         $MC_k \leftarrow mean_L(S_C)$;
21:         $k \leftarrow (k\ modulo\ H) + 1$;
22:     $t \leftarrow t + 1$;
```

III. Evaluating Alternative Methods for Sampling the Success-History

As described in Sec. II, the standard method for using the success history during search is to first randomly select an index $i \in [1, H]$ (H is the size of the success history array), and use the i'th element of MF and MC to generate a (F, C) pair. Below, we consider a simple, alternative sampling strategy, with some surprising results.

A. Evaluating a Fixed Sampling Strategy

We first consider a *fixed sampling strategy*, which in Alg. 2, line 7, always sets r_i to a fixed index value a, i.e., always pick the a'th elements from the MF and MC arrays.

We evaluate the fixed sampling strategy using functions F1∼F30 from the CEC2014 benchmark set [12], for dimensionality $D = 10$ and 30. We ran 48 trials per algorithm configuration per function, with $10,000 \times D$ fitness evaluations per trial. The fixed samping positions evaluated were $a \in \{1, 5, 10, 15, 20, 30, 60, 90\}$ for $D = 30$, and $a \in \{1, 3, 5, 7, 10\}$ for $D = 10$. The success history sizes were $H = 10$ for $D = 10$, and $H = 90$ for $D = 30$. Although this differs from the standard setting of $H = D/2$, this allows us to observe the effect of large values of a.

Figure 1 shows the cumulative probability of achieving target fitness (y) vs. number of fitness evaluations (x), for CEC2014 problems on $D = 10$ and $D = 30$ ($48 \times 30 = 1440$ trials total). The target fitness is the median fitness achieved among all trials for each problem ($48 \times k$), where the k, the number of fixed values evaluated was 8 for $D = 30$, 5 for $D = 10$. It can be seen that the performance varies significantly depending on the fixed sampling location a, particularly for $D = 30$. In particular, $a = 15$ (for $D = 30$) and $a = 5$ (for $D = 10$) performed well. Interestingly, these successful a-values correspond to $D/2$.

[1]An earlier algorithm, SaDE [11] adapted its C parameter values using a similar historical memory, but unlike SHADE, which stores per-generation means of successful F and C, SaDE explicitly stored all successful C parameters

Algorithm 3 EnJADE

```
 1: $t \leftarrow 1$, Initialize population;
 2: Initialize contents of $MF$ and $MC$ arrays;
 3: $k \leftarrow 1$;
 4: while not termination condition do
 5:     $S_F \leftarrow \emptyset, S_C \leftarrow \emptyset$;
 6:     for $i = 1$ to $N$ do
 7:         $F_{i,t} = randc(MF_{\mathbf{k}}, 0.1)$;
 8:         $C_{i,t} = randn(MC_{\mathbf{k}}, 0.1)$;
 9:         generate $v_{i,t}$ using current-to-pbest mutation;
10:         generate $u_{i,t}$ using binomial crossover;
11:     for $i = 1$ to $N$ do
12:         if $f(u_{i,t}) \leq f(x_{i,t})$ then
13:             $x_{i,t+1} \leftarrow u_{i,t}$;
14:             $S_F \leftarrow S_F \cup F_{i,t}, S_C \leftarrow S_C \cup C_{i,t}$;
15:         else
16:             $x_{i,t+1} \leftarrow x_{i,t}$;
17:     if $S^F, S^C \neq \emptyset$ then
18:         $MF_k \leftarrow mean_L(S_F)$;
19:         $MC_k \leftarrow mean_L(S_C)$;
20:         $k \leftarrow (k \ modulo \ H) + 1$;
21:     $t \leftarrow t + 1$;
```

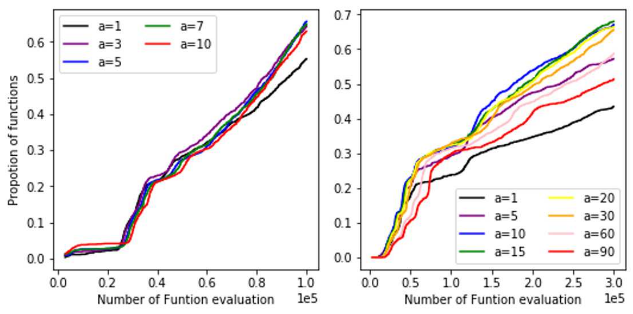


Fig. 1: Cumulative probability of achieving target fitness (y) vs. number of fitness evaluations (x), for CEC2014 problems, $D = 10$ and $D = 30$ dimensions ($48 trials \times 30 problems = 1440$ trials per algorithm). The target fitness is the median fitness achieved among all trials for each problem ($48 trials/algorithm \times k$), where the k, the number of fixed values evaluated was 8 for $D = 30$, 5 for $D = 10$.

To assess whether the overall peformance differences indicated in Figure 1 are statistically significant, the left side of Table I compares, for each problem in F1∼F30, the best fitness values found on each trial using the fixed sampling index $a = 15$ vs. other values of a, for $D = 30$, using the Wilcoxon rank sum test. Similarly, the right side of Table I compares $a = 5$ vs. other values of a for $D = 10$. Table I shows that overall, using the fixed index $a = D/2$ performs well compared to other fixed index a-values.

As explained in Sec. II, JADE can be seen as a special case of SHADE with history size $H = 1$, which is also equivalent to SHADE which always uses the first ($a = 1$) element of the success history. Our data shows that always using $a = 1$ is significantly worse than always using $a = D/2$, i.e., greedily

	$D = 30$ dimensions								$D = 10$ dimensions				
a=	1	5	10	15	20	30	60	90	1	3	5	7	10
F1	≈	≈	≈		≈	≈	-	-	≈	≈		≈	≈
F2	≈	≈	≈		≈	≈	≈	≈	≈	≈		≈	≈
F3	≈	≈	≈		≈	≈	≈	≈	≈	≈		≈	≈
F4	-	≈	≈		≈	≈	≈	≈	≈	≈		≈	≈
F5	-	-	≈		≈	-	-	-	≈	≈		≈	≈
F6	-	-	-		≈	+	+	+	-	≈		≈	-
F7	-	≈	≈		≈	≈	≈	≈	-	≈		≈	≈
F8	≈	≈	≈		≈	≈	≈	≈	≈	≈		≈	≈
F9	≈	+	≈		≈	≈	≈	≈	≈	≈		≈	≈
F10	-	≈	≈		≈	≈	-	-	-	≈		≈	≈
F11	-	-	≈		≈	+	≈	-	-	≈		≈	≈
F12	-	-	≈		≈	-	-	-	≈	≈		≈	≈
F13	-	≈	≈		≈	≈	≈	-	≈	≈		≈	≈
F14	-	-	≈		-	≈	-	-	-	≈		≈	≈
F15	-	≈	≈		≈	≈	≈	-	≈	≈		-	-
F16	-	-	≈		≈	-	-	-	≈	≈		+	≈
F17	≈	≈	+		≈	≈	≈	≈	≈	≈		≈	≈
F18	-	-	≈		≈	≈	≈	≈	-	≈		≈	≈
F19	-	-	≈		≈	+	+	≈	-	-		≈	≈
F20	-	-	-		≈	≈	+	+	-	≈		≈	+
F21	≈	≈	≈		≈	≈	≈	≈	≈	≈		≈	-
F22	+	≈	≈		≈	≈	≈	≈	≈	≈		≈	≈
F23	≈	≈	≈		≈	≈	≈	≈	≈	≈		≈	≈
F24	-	-	≈		≈	+	≈	+	≈	≈		≈	≈
F25	≈	≈	+		+	≈	≈	≈	-	≈		≈	≈
F26	-	≈	+		≈	≈	-	-	+	≈		≈	≈
F27	-	-	-		≈	≈	≈	+	≈	≈		≈	≈
F28	-	-	≈		≈	≈	≈	≈	-	≈		≈	≈
F29	-	≈	≈		≈	≈	≈	-	≈	≈		≈	≈
F30	-	-	≈		≈	+	+	≈	-	≈		-	-

TABLE I: Comparison of fixed success-history sampling strategies (always using the a-th element in the history) for CEC2014 problems, $D = 30$ and $D = 10$ dimensions. The Wilcoxon ranked sum test results （+: better, -: worse, ≈: no significant difference） are shown for comparisons between $D/2$ and other values of a (for $D = 30$, comparisons vs. $a = 15$; for $D = 10$, comparisons are vs. $a = 5$).

continuing to generate parameter values similar to the most recent successful parameter values as JADE does is not a particularly good strategy in this context.

B. Sampling a subrange of the history around a index a

The previous experiment showed that a fixed sampling strategy using the $D/2$-th element in the success histories performed well compared to other fixed sampling strategies. Next, we consider sampling indices *around* the $D/2$-th element – i.e., we test whether sampling randomly from a range more restricted than SHADE is helpful.

We used the same experimental settings as the previous experiment, but instead of a fixed strategy which always used the a-th element of the success histories, each sample (Alg. 2, line 7) is from a range r centered around $D/2$, where the width of the ranges is varied. For $D = 30$ dimensions, we evaluated $r = 10 \sim 15, 13 \sim 15, 14 \sim 15, 15, 15 \sim 16, 15 \sim 17$ と $15 \sim 20$. For $D = 15$ dimensions, we evaluated $r = 1 \sim 5, 4 \sim 5, 5, 5 \sim 6, 5 \sim 10$.

Figure 2 shows the cumulative probability of achieving target fitness (y) vs. number of fitness evaluations (x), for CEC2014 problems on $D = 10$ and $D = 30$ ($48 \times 30 = 1440$ trials total). The target fitness is the median fitness achieved among all trials for each problem ($48 \times k$), where the k, the number of ranges evlauated was 7 for $D = 30$, 5 for $D = 10$. For $D = 10$, there is no clear difference among the sampling ranges. On the other hand, for $D = 30$, there are significant differences in overall performances among the sampling ranges. Notably, the fixed $a = 15$ strategy clearly outperforms all of the other sampling ranges – there does not appear to be an advantage to sampling a wider range than the fixed $a = 15$ strategy.

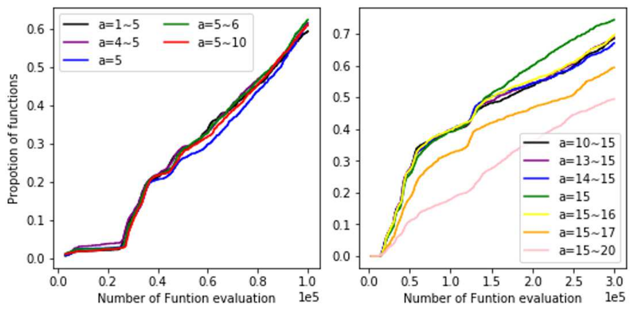


Fig. 2: Cumulative probability of achieving target fitness (y) vs. number of fitness evaluations (x), for CEC2014 problems on $D = 10$ and $D = 30$ ($48 trials \times 30 problems = 1440$ trials total). The target fitness is the median fitness achieved among all trials for each problem ($48 trials/algorithm \times k$), where the k, the number of ranges used for sampling the success history values evaluated was 7 for $D = 30$, 5 for $D = 10$.

IV. Ensemble JADE: Reinterpreting Fixed Index Success History Sampling as Cyclic Allocation of Resources Among an Ensemble

In Sec. II, we noted that JADE can be seen as a special case of SHADE with history size $H = 1$, and furthermore, we noted in Sec. III, JADE can also be seen as a variant of SHADE which in Alg. 2, Line 7 deterministically chooses the 1st element. We also observed experimentally in Sec. III that a variation of SHADE which deterministically selects the $H/2$-th index in Alg. 2, Line 7 performs quite well (better than the standard uniform sampling from H). It turns out that we can also reinterpret this deterministic variant of SHADE in terms of JADE.

Ensemble JADE (EnJADE), shown in Alg. 3, is a variant of JADE which, instead of a single pair of μ_F and μ_C parameters, has an arrays $MF = (MF_1, \cdots, MF_H)$ and $MC = (MC_1, \cdots, MC_H)$. These arrays store MF_i and MC_i values which are used to generate $F_{i,t}$ and $C_{i,t}$ values in lines 7-8. Unlike SHADE in Alg. 2, which samples uniformly randomly from M^F and M^C arrays, EnJADE deterministically accesses the k-th element, where k is simply incremented at each iteration (modulo H). In other words, EnJADE has an ensembles of parameters MF and MC, and cycles deterministically, sequentially using each of the ensemble members in turn. This is exactly equivalent to the deterministic variant of SHADE which always chooses the H-th elements of the history arrays.[2]

Note that we are *not* claiming that EnJADE is a particularly novel algorithm. The significane of EnJADE is the new perspective provided by considering MF and MC *not* as a "history" which implies the importance of temporal ordering, but as H completely independent members of a parameter ensemble among which the search allocates effort. enabling a new, simple intuitive understanding of its behavior. It is not clear that "Maintaining a history and always using the H-th index" has an obvious intuitive motivation/interpretation – why H, as opposed to some other index? In contrast, "cycling among independent H sets of parameters" has an obvious intuitive interpretation – In contrast to standard JADE, which relies on a single μ_F, μ_C parameter pair and may fail if μ_F and μ_C are inappropriate for the given search space, EnJADE diversifies search among H different μ_F and μ_C pairs, allocating equal effort to each such pair in a round-robin manner.

Importantly, we can also reinterpret SHADE (with random history sampling) in terms of the new framework provided by EnJADE. Original SHADE can be viewed as a variant of EnJADE, but instead of allocating resources equally among the ensemble elements equally in a round-robin manner (1 member selected each generation), original SHADE schedules effort in randomized order, where many members are selected each generation (each individual uses a possibly different element, Alg. 2, line 7. If this ordering (round-robin/cyclic vs. randomized) was the only difference between original SHADE and EnJADE, there should not be such a significant difference between original SHADE vs. EnJADE (SHADE with determinsitic history sampling), as in the long-run, randomized sampling and round-robin allocation (1 member used each generation) should converge to roughly the same allocation.

In addition to the order in which the ensemble members are selected/sampled, another, important difference between original SHADE and EnJADE is *the index of the ensemble members which are updated after trial vector generation*. EnJADE selects (round-robin) the k-th pair MF_k and MC_k from the ensemble, uses them to generate (F, C) pairs which are then used for trial vector generation, and then, the Lehmer means of successful pairs are used to update MF_k and MC_k, i.e., each generation, we update the k-th pair in the ensemble based on successes in that generation of the previous values in that k-th pair.

In contrast, original SHADE randomly selects for each individual the r_i'th pair from the ensemble (history) and uses the pair to generate (F, C) pairs used for trial vector generation. However, instead of updating the r_i-th element of the ensemble/history, the k-th element (where k incremented per generation cyclically from $1.,,,H$) is updated. Thus, in original SHADE, the ensemble/history element (k-th) which is updated/overwritten at the end of the generation is not likely to be strongly related to the (r_i-th) ensemble/history elements used to generate (F, C) values during the generation.

V. Experimental Evaluation of EnJADE

We experimentally compared the performance of the following configurations of JADE and SHADE. The implementation of SHADE evaluated is SHADE 1.1.1, the code from the original author of SHADE [14]. All of the other evaluated algorithms were implemented by modifying SHADE 1.1.1.

- JADE: Population 100, archive rate 2 and *pbest* rate 0.1.
- SHADE: history sizes $H = 15$. Other parameters are the same as JADE.

[2]The use of an ensemble of parameters is related to [13], but EPSDE does not update its pool of F and C values.

- SHADE with Restarts (R4): Same as SHADE, except that R4 restarts 4 times (every 300,000/4 generations).
- EnJADE: Ensemble JADE with ensemble size 15.

R4 is a simple control configuration to test whether the proposed ensemble methods perform differently than simply restarting the search at fixed intervals (we also tried several other restart configurations and this R4 was the best-performing among them).

Table II compares the mean best fitnesses found by each of the algorithms/configurations above vs. EnJADE, according to the Wilcoxson ranked-sum test （$p = 0.05$; +: better than EnJADE, -: worse than EnJADE, $\approx$: no significant difference）.

Figure 3 compares the overall performance of EnJADE, SHADE (H15, corresponds to the standard $H = D/2$ setting), SHADE with Restarts and JADE, on $D = 30$ dimensional problems, according to the cumulative probability of achieving a target fitness, where the target fitness is the median fitness achieved among all trials for each problem ($48 \times 4 algorithms$).

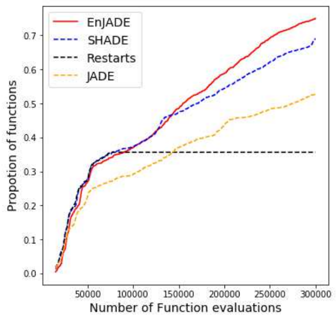


Fig. 3: Comparison of EnJADE, SHADE, and JADE. Cumulative probability of achieving target fitness (y) vs. number of fitness evaluations (x), for CEC2014 problems, $D = 30$ dimensions ($48 trials \times 30 problems = 1440$ trials per algorithm). The target fitness is the median fitness achieved among all trials for each problems ($48 trials/algorithm \times 4 algorithms = 192$).

Overall, Figure 3 and Table II show that EnJADE outperforms SHADE, JADE, and SHADE with restarts. Table II shows that EnJADE tends to outperform SHADE on the multimodal and hybrid functions (F12$\sim$F30). SHADE with restarts performs worse than SHADE and EnJADE. Thus, adding an ensemble to JADE (EnJADE) seems to result in better performance overall than success history and restarts (SHADE with restarts).

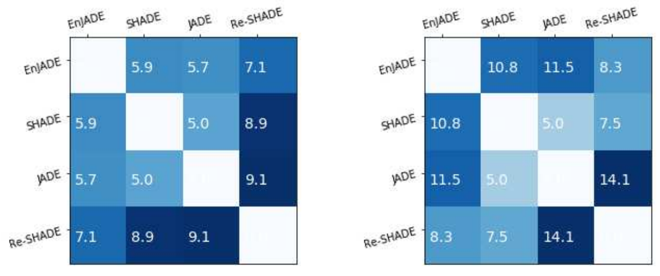
A. Analysis of Search Behavior

In addition to the search performance, we analyzed the difference in search behavior, as indicated by the distribution of (F, C) parameter values generated by the parameter adaptation mechanisms during search. In the same experimental runs as the performance comparison below, we collected all (F, C) values for all algorithms, all problems, and all trials. From this, we obtained sample probability distributions $P_{a,i}$ for algorithm configuration a and problem i (data from all 48 trials is included in $P_{a,i}$.

TABLE II: Comparison of EnJADE(H=15), JADE(H=1), SHADE(H=15), and SHADE(H=15, 4 times a quarter evaluations) with Restarts on CEC14 benchmark problems (D=30 dimensions). Mean of best objective function values found (48 runs) The +/-/$\approx$ symbols indicate result of Wilcoxon ranked-sum est vs. our proposed method, EnJADE. (+: better than EnJADE, -: worse than EnJADE, $\approx$: no significant difference).

	EnJADE	JADE		Restarts		SHADE	
F1	4.39E+2	$\approx$	2.02E+2	-	2.14E+4	$\approx$	1.39E+2
F2	0.00	$\approx$	0.00	$\approx$	0.00	$\approx$	0.00
F3	0.00	$\approx$	0.00	$\approx$	0.00	$\approx$	0.00
F4	0.00	$\approx$	2.64	-	7.47	$\approx$	1.32
F5	2.01E+1	-	2.02E+1	-	2.04E+1	-	2.01E+1
F6	2.59E-1	-	1.62	-	8.41E-1	-	9.19E-1
F7	1.54E-4	-	2.21E-3	$\approx$	3.08E-4	$\approx$	3.59E-4
F8	0.00	$\approx$	0.00	-	1.41E+1	$\approx$	0.00
F9	1.74E+1	$\approx$	1.83E+1	-	5.37E+1	+	1.37E+1
F10	6.94E-3	+	3.04E-3	-	3.15E+2	$\approx$	5.21E-3
F11	1.31E+3	-	1.50E+3	-	3.10E+3	$\approx$	1.43E+3
F12	1.15E-1	-	2.13E-1	-	5.52E-1	-	1.57E-1
F13	1.90E-1	-	2.13E-1	-	2.29E-1	-	2.13E-1
F14	2.38E-1	-	2.60E-1	-	2.53E-1	$\approx$	2.46E-1
F15	2.50	-	2.76	-	6.27	$\approx$	2.50
F16	8.13	-	9.22	-	1.05E+1	-	9.16
F17	1.10E+3	$\approx$	1.07E+3	$\approx$	1.06E+3	$\approx$	1.03E+3
F18	5.48E+1	-	8.04E+1	-	6.53E+1	$\approx$	5.07E+1
F19	3.09	-	5.50	-	5.39	-	4.60
F20	1.04E+1	-	1.52E+1	-	2.11E+1	-	1.44E+1
F21	2.76E+2	$\approx$	3.14E+2	$\approx$	2.54E+2	$\approx$	2.74E+2
F22	1.19E+2	$\approx$	8.85E+1	-	2.15E+2	$\approx$	9.96E+1
F23	3.15E+2	$\approx$	3.15E+2	$\approx$	3.15E+2	$\approx$	3.15E+2
F24	2.25E+2	-	2.33E+2	-	2.27E+2	-	2.27E+2
F25	2.03E+2	$\approx$	2.03E+2	$\approx$	2.03E+2	$\approx$	2.03E+2
F26	1.02E+2	-	1.00E+2	-	1.00E+2	$\approx$	1.00E+2
F27	3.29E+2	-	3.68E+2	-	3.29E+2	-	3.29E+2
F28	8.31E+2	-	8.64E+2	-	8.47E+2	-	8.48E+2
F29	7.25E+2	$\approx$	7.16E+2	-	7.57E+2	$\approx$	7.24E+2
F30	1.80E+3	-	2.56E+3	$\approx$	1.93E+3	$\approx$	1.98E+3

We computed $KL(P_{a,i}, P_{b,i})$, the Kullback-Leibler divergence between distributions $P_{a,i}$ vs. $P_{b,j}$, for each problem, between algorithm pairs a and b. The KL-divergence $KL(P_{a,i}, P_{b,i})$ indicates the similarity in the distribution of (F, C) values in $P_{a,i}$ and $P_{b,i}$ and is therefore a rough indicator of the difference in search behavior between algorithms a and b on problem i. We further compute the sum of these KL-divergences between algorithms a and b across all problems, $\sum_{i \in F1,\dots,F30} KL(P_{a,i}, P_{b,i})$, which indicates the overall difference in search behavior between algorithms a and b on the 30 problems in the CEC14 benchmark set.



(a) (b)

Fig. 4: Comparison of (F, C) values generated by EnJADE, JADE, SHADE, and SHADE with Restarts (“Re-SHADE”). The numbers indicate the sum of the KL-divergence between the distribution of F and C values generated during search.

Figure 4 shows pairwise comparisons of $\sum_{i \in F1,\dots,F30} KL(P_{a,i}, P_{b,i})$ for both F and C. The differences between EnJADE vs. SHADE and EnJADE vs. JADE are much bigger than SHADE vs. JADE, indicating that although EnJADE can be viewed as a small modification to SHADE, the behavior is quite different. Overall, for both F and C, SHADE with restarts has the largest divergence from SHADE and JADE.

VI. Ensembles and Population Reduction

LSHADE [9] is a variant of SHADE which reduces the size of the population during search according to a linear reduction schedule. The intuitive motivation for population reduction is that initially, the search uses a standard population size to avoid committing too early to one area of the search space (possible traps), but as the search progresses, search is intensified by reducing population size so that it searches for a (local) optimum in a smaller region.

LSHADE has been shown to perform significantly better than SHADE, and most of the SHADE variants which have ranked highly on recent CEC competitions have been based on LSHADE. As population reduction is orthogonal to the parameter adaptation mechanism, EnJADE can be straightforwardly to LEnJADE (EnJADE with linear populatoin reduction). Thus, we evaluate our proposed ensemble scheme with population reduction, to verify whether the performance improvement observed in the previous section can also be obtained when the base algorithm is more competitive.

We evaluate LEnJADE using the same population reduction schedule as in [9]. For LSHADE, we use the LSHADE 1.0.1 C++ code by the original author of LSHADE [15]. LEnJADE was implemented by modifying LSHADE 1.0.1. We used the same CEC14 benchmarks ($D = 10, 30$ dimensions) and experimental settings as in Sec. V and compared (1) LSHADE: success history sizes $H = 15$. Population 100, archive rate 1.4 and *pbest* rate 0.11. (2) LJADE: same parameters as LSHADE. (3) LEnJADE: Ensemble JADE with ensemble size 15.

Table III compares the mean best fitnesses found by each of the algorithms above vs. LJADE, according to the Wilcoxson ranked-sum test （$p = 0.05$; +: better than LSHADE, -: worse than LSHADE, $\approx$: no significant difference）.

Figure 5 compares the overall performance of LEnJADE, as well as EnJADE, LSHADE, and SHADE (EnJADE, LSHADE, and SHADE data are reused from the experiment in Sec. V), according to the cumulative probability of achieving a target fitness, where the target fitness is the median fitness achieved among all trials for each problems ($48 \times \times 4 algorithms$).

Figure 5 and Table III show that LEnJADE has the best overall performance among all of the evaluated algorithms.

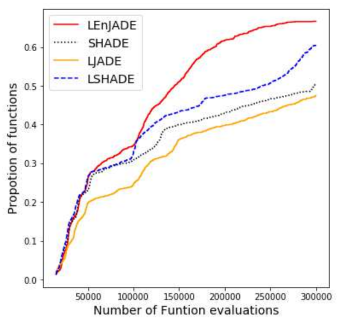


Fig. 5: Comparison of LEnJADE, LSHADE, and LJADE. Cumulative probability of achieving target fitness (y) vs. number of fitness evaluations (x), for CEC2014 problems, $D = 30$ dimensions ($48 trials \times 30 problems = 1440$ trials total per algorithm). The target fitness is the median fitness achieved among all trials for each problem ($48 trials/algorithm \times 4 algorithms = 192$ trials).

TABLE III: Comparison of LEnJADE vs. LSHADE (CEC2014 benchmarks, $D = 30$ dimension). Mean of best objective function values found (48 runs) The +/-/$\approx$ symbols indicate result of Wilcoxon ranked-sum test vs. our proposed method LEnJADE (p=0.05) (+: better, -: worse, $\approx$: no significant difference).

	LEnJADE	LJADE	LSHADE		LEnJADE	LJADE	LSHADE
F1	5.13E+02	+ 3.52E+02	+ 4.40E+02	F16	8.53	$\approx$ 8.48E+00	- 8.79
F2	0	$\approx$ 0	$\approx$ 0	F17	1.05E+03	$\approx$ 1.13E+03	$\approx$ 1.05E+3
F3	0	$\approx$ 0	$\approx$ 0	F18	5.40E+01	- 6.45E+01	$\approx$ 5.91E+1
F4	1.35	$\approx$ 6.6	$\approx$ 0	F19	3.18	- 4.77E+00	- 4.01
F5	2.00E+01	- 2.00E+01	- 2.00E+01	F20	1.07E+01	- 1.77E+01	- 1.79E+1
F6	3.50E-01	- 2.9	- 1.40E+00	F21	2.35E+02	$\approx$ 2.60E+02	$\approx$ 2.52E+2
F7	2.06E-04	- 2.92E-03	$\approx$ 5.14E-04	F22	9.76E+01	$\approx$ 8.46E+01	$\approx$ 8.95E+1
F8	0	$\approx$ 0	$\approx$ 0	F23	3.15E+02	$\approx$ 3.15E+02	$\approx$ 3.15E+2
F9	1.80E+01	+ 1.55E+01	+ 1.33E+01	F24	2.25E+02	- 2.32E+02	- 2.27E+2
F10	1.30E-02	+ 3.90E-03	$\approx$ 1.82E-02	F25	2.03E+02	+ 2.03E+02	$\approx$ 2.03E+2
F11	1.39E+03	$\approx$ 1.33E+03	$\approx$ 1.40E+03	F26	1.00E+02	- 1.00E+02	- 1.00E+2
F12	4.79E-02	- 1.08E-01	- 9.03E-02	F27	3.24E+02	- 3.68E+02	- 3.25E+2
F13	1.46E-01	- 1.99E-01	- 1.88E-01	F28	8.27E+02	- 8.56E+02	- 8.41E+2
F14	2.28E-01	- 2.60E-01	- 2.44E-01	F29	7.28E+02	$\approx$ 7.25E+02	$\approx$ 7.28E+2
F15	2.41	+ 2.11E+00	+ 1.92E+00	F30	1.49E+03	- 2.17E+03	$\approx$ 1.65E+3

VII. Success Histories and Robustness

We have shown above that on the standard CEC2014 benchmarks, the use of success histories is not necessary in order to improve upon the performance of JADE, and that the performance benefits from the diversity provided by the success history mechanism in SHADE can be achieved (and even exceeded in some cases) by applying a straightforward ensemble mechanism to JADE. A natural question is whether the success history mechanism can be entirely replaced by ensembles, or whether there are some cases where success histories offer a clear advantage compared to ensembles.

One possible advantage of success histories over ensembles of distributions is robustness during search. In JADE/SHADE and their variants, the parameters controlling F, C generation are updated when successful trial vectors are generated. However, in some situations, it is possible that such a success is "accidental", e.g., in problems with noisy fitness functions, a "bad" (F, C) might result in a successful trial generation.

We evaluated the robustness of EnSHADE, SHADE, and JADE, by injecting noise as follows, every N generations ($N = 100$). instead of inserting the successful $mean_L(S_F)$ and $mean_L(S_C)$ into sets MF_k and MC_k in Alg. 2, lines 19-20 (and similarly for JADE and EnJADE), we inject extreme F and C values, $F_{ext} \rightarrow MF_k, C_{ext} \rightarrow MC_k$. In the "NL" noise setting, F_{ext} and C_{ext} are 0.1. In the "NH" noise setting, F_{ext} and C_{ext} are 0.9.

Each column in Table IV compares the result (best objective function values found, 48 runs) of an algorithm configuration with artificial noise injection vs. the noiseless algorithm, e.g., The (SHADE, population 20, NL) column compares SHADE with population 20, NL noise injection vs. SHADE with population 20. Comparison results are according to the Wilcoxson ranked-sum test （$p = 0.05$; +: noisy configuration performed better than standard configuration, -: noisy configuratoin performed worse than standard configuration, $\approx$: no significant difference. For both population sizes 20 and 100, as well as both NL and NH noise, the performance of SHADE is affected less by the injected noise, compared to JADE and EnJADE, as indicated by the larger number of "$\approx$" results. This shows that maintaining a discrete success history results in more stable/robust behavior than maintaining a smaller set of distribution means. However, note that in some cases, noise improves performance, as indicated by the '+' results, so although stability is often desirable, it does not necessarily mean "better performance". Intentional noise injection aimed to improve performance is a direction for future work.

TABLE IV: Evaluating the stability/robustness of EnJADE, JADE, and SHADE: Each column compares the result (best objective function values found, 48 runs) vs. the noiseless algorithm, e.g., the (SHADE, population 20, NL) column compares SHADE with population 20, NL noise injection vs. SHADE with population 20. Comparison according to Wilcoxson ranked-sum test （$p = 0.05$; +: noisy configuration performed better than standard configuration, -: noisy configuratoin performed worse than standard configuration, $\approx$: no significant difference）. Noise is injected every 100 generations. NL: inject F,C=0.1, NH: inject F,C=0.9.

	population size 20						population size 100					
	EnJADE		JADE		SHADE		EnJADE		JADE		SHADE	
noise	NL	NH	NL	NH	NL	NH	NL	NH	NL	NH	NL	NH
$\approx$	24	21	24	18	**30**	**25**	**24**	22	22	20	22	**28**
+	5	1	6	2	0	1	0	2	3	6	0	1
-	1	8	0	10	0	4	6	6	5	4	8	1
F1	$\approx$	+	s	$\approx$	$\approx$	$\approx$	-	-	-	+	-	$\approx$
F2	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$
F3	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$
F4	+	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$
F5	$\approx$	-	$\approx$	-	$\approx$	-	-	$\approx$	$\approx$	+	$\approx$	$\approx$
F6	$\approx$	-	+	-	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	+	$\approx$	$\approx$
F7	$\approx$	$\approx$	+	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$
F8	+	-	$\approx$	-	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	-	$\approx$	$\approx$
F9	$\approx$	-	$\approx$	-	$\approx$	$\approx$	$\approx$	-	$\approx$	-	-	$\approx$
F10	$\approx$	-	$\approx$	-	$\approx$	$\approx$	$\approx$	-	+	-	$\approx$	$\approx$
F11	$\approx$	-	$\approx$	-	$\approx$	-	$\approx$	$\approx$	$\approx$	-	-	$\approx$
F12	$\approx$	$\approx$	+	-	$\approx$	-	-	+	$\approx$	+	$\approx$	$\approx$
F13	+	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	-	$\approx$	$\approx$	+	-	$\approx$
F14	$\approx$	-	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	-	$\approx$
F15	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	-	$\approx$	-	$\approx$	$\approx$	-
F16	+	-	$\approx$	-	$\approx$	-	-	-	-	$\approx$	$\approx$	$\approx$
F17	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	+	$\approx$	$\approx$	$\approx$
F18	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	+	$\approx$	-	$\approx$
F19	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	+	$\approx$	$\approx$
F20	$\approx$	$\approx$	+	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$
F21	$\approx$	$\approx$	+	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$
F22	$\approx$	$\approx$	$\approx$	-	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	+
F23	$\approx$	$\approx$	+	+	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$
F24	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	-	$\approx$	$\approx$	$\approx$	$\approx$
F25	$\approx$	$\approx$	$\approx$	-	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$
F26	-	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	+	-	+	-	$\approx$
F27	+	$\approx$	$\approx$	$\approx$	$\approx$	+	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$
F28	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$
F29	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	-	$\approx$	-	$\approx$
F30	$\approx$	$\approx$	$\approx$	+	$\approx$	$\approx$	$\approx$	-	$\approx$	$\approx$	$\approx$	$\approx$

VIII. Conclusion

Despite the success of SHADE-based adaptive DE and proliferation of SHADE variants, its main, distinguishing feature, the success history, has not been well-understood, and most SHADE variants continue to use the success history in the same way as the original SHADE. We reinvestigated the use of success history, and showed that:

(1) The standard approach of uniform random sampling from the history does not result in the best performance – sampling from a fixed location in the history performs surprisingly well. (2) This fixed sampling strategy is equivalent to and can be reinterpreted more simply as EnJADE, which is JADE with an ensemble of μ_C and μ_F pairs, among which effort is allocated equally in a round-robin strategy (1 member per generation), and the ensemble member used in that generation is updated. (3) EnJADE is competitive with SHADE; performance continues to be competitive when linear population reduction is used. (4) With respectness of robustness/stability of search behavior, which was the original motivation given for the success history [7], SHADE performance was more stable than JADE and EnJADE when noise was artificially injected into the parameter adaptation mechanisms.

Thus, we have shown that the "history" arrays used by SHADE can be reinterpreted and used with a different sampling/update policy, resulting in significant improvement compared to SHADE. Although it seems that the update policy (updating the same element which is used during that generation to generate F, C pairs) is likely important, it is less clear whether round-robin, equal allocation of sampling is important. More in-depth investigations of the update and sampling policies is a direction for future work.

References

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

[2] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Art. Intell. Rev.*, vol. 33, no. 1-2, pp. 61–106, 2010.

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

[4] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution - an updated survey," *Swarm and Evol. Comput.*, vol. 27, pp. 1–30, 2016.

[5] R. Tanabe and A. Fukunaga, "Reviewing and benchmarking parameter control methods in differential evolution," *IEEE Transactions on Cybernetics*, vol. 50, no. 3, pp. 1170–1184, 2020.

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

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

[8] ——, "How Far Are We from an Optimal, Adaptive DE?" in *PPSN*, 2016, pp. 145–155.

[9] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *IEEE CEC*, 2014, pp. 1658–1665.

[10] J. Tvrdık, "Competitive Differential Evolution," in *MENDEL*, 2006, pp. 7–12.

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

[12] 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 Univ. and Nanyang Technological Univ., Tech. Rep., 2013.

[13] 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, pp. 1679–1696, 2011.

[14] R. Tanabe, "SHADE version 1.1.1." [Online]. Available: https://ryojitanabe.github.io/code/SHADE1.1.1_CEC2014_c++.zip

[15] ——, "L-SHADE version 1.0.1." [Online]. Available: https://ryojitanabe.github.io/code/LSHADE1.0.1_CEC2014.zip