Tuning Differential Evolution for Cheap, Medium, and Expensive Computational Budgets

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

***Abstract*—This paper presents a parameter tuning study of Differential Evolution (DE) algorithms, including both standard DE as well as variants of the state-of-the-art adaptive DE, SHADE for both cheap and expensive optimization scenarios. Using the algorithm configuration tool SMAC, the DE variants are tuned independently for three different scenarios: expensive ($10^2 \times D$ evaluations), medium ($10^4 \times D$ evaluations), cheap ($10^5 \times D$ evaluations), where D is the benchmark problem dimensionality. Each of these tuned parameter settings is then tested under both cheap and expensive scenarios, which enables us to analyze the effect of both the tuning and test scenario on the performance of the tuned algorithm. We evaluate restarting variants of DE (R-DE), as well as restarting variants of SHADE (R-SHADE) and L-SHADE (RL-SHADE). For the parameter tuning phase, we use the CEC2014 benchmarks as training problems, and for the testing phase, we use all 24 problems from the BBOB benchmark set. We also compare these DE variants with state-of-the-art restart CMA-ES variants (HCMA, BIPOP-CMA-ES, and IPOP-CMA-ES). For both cheap and expensive scenarios, DE algorithms perform very well for low-dimensional problems. In particular, for the expensive scenario, the simple, restarting DE (R-DE) performs quite well, and on the cheap scenario, RL-SHADE performs well.**

I. Introduction

In the Evolutionary Computation (EC) community, empirical studies of Evolutionary Algorithms (EAs) have tended to be based on a relatively large budget of fitness evaluations. For example, Yao's 13 classical benchmarks [1] prescribe $1.5 \times 10^4 \sim 2.0 \times 10^6$ evaluations for 30-dimensional problems, the CEC2005/2014 benchmarks [2], [3] use $10^4 \times D$ evaluations (where D is the dimensionality of the problem), and the SOCO benchmarks [4] use $5,000 \times D$ evaluations. However, some real-world applications of EAs require executing very expensive simulations (up to 10 minutes/run) in order to evaluate the fitness of a single individual [5].

Thus, in recent years, there has been much research on such *expensive optimization problems* in the EC community. In contrast to traditional benchmark settings, evaluating the fitness of thousands or hundreds of individuals is infeasible for problems – $1,000$ or so evaluations is a more realistic limit on the number of fitness evaluations. The standard approach for such expensive optimization problems use *surrogate models*, in which a (fast) proxy model for the expensive optimization function is learned and used instead of the actual fitness function [5]. In this surrogate approach, the number of "fitness evaluations" performed by the evolutionary algorithms is quite large (e.g., 10^4 – 10^5), but most of these evaluations are performed using the surrogate function, and only a small number (e.g., $< 1,000$) of calls to the actual fitness function are made.

It is widely known that in general, the performance of an EA depends on its control parameter settings, and there is a large body of work related to parameter tuning [6]. However, most of the previous work is based on *cheap* scenarios using at least tens of thousands of fitness evaluations, and there has been relatively little work on parameter tuning for expensive scenarios. Cáceres et al have recently shown that when ACO is tuned specifically for expensive scenarios, the tuned parameters significantly differ from standard control parameter values, and are much more "greedy" than standard values [7]. Liao et al have tuned IPOP-CMA-ES [8] for expensive scenarios [9] and standard (cheap) scenarios [10] separately. Although it is not explicitly pointed out by Liao et al, the population size, and initial step size of IPOP-CMA-ES obtained by tuning for expensive scenarios is significantly smaller (greedier) than those for the standard cheap scenario. Thus, control parameters that are appropriate for standard, cheap scenarios are not necessarily appropriate for expensive scenarios, and it is necessary to tune parameters depending on the computational budget.

In this paper, we present a parameter tuning study for variants of Differential Evolution (DE) [11]. By performing tuning for several different computational budgets, including expensive optimization scenarios, we investigate the effect of the budget used during training on the tuned parameters. DE is an EA that was primarily designed for real parameter optimization problems [11]. Despite its relative simplicity, DE has been shown to be competitive with more complex optimization algorithms, and has been applied to many practical problems [12]. As with other EAs, the performance of DE is greatly affected by its control parameters, so there have been numerous, previous parameter studies of DE [11], [13], [14]. Furthermore, many *adaptive* variants of DE that adapt their control parameter settings while solving a problem have been investigated [15], [16], [17].

However, to our knowledge, there have been no previous, thorough studies of DE that includes expensive scenarios. For example, the widely cited parameter studies of DE by Gämperle et al [13] and Montes et al [14] use a budget of 1.5×10^5, 4.0×10^5 evaluations for $2 \sim 20$ dimensions, and 1.2×10^5 evaluations for 30 dimensions, respectively. Thus, there is need for a parameter study which includes expensive scenarios that are an important class of problems in practice.

```
    // Initialization phase
1   $G = 1$, $N_G = N^{init}$, Archive $\boldsymbol{A} = \varnothing$;
2   Initialize population $\boldsymbol{P}_G = (\boldsymbol{x}_{1,G}, ..., \boldsymbol{x}_{N,G})$ randomly;
3   Set all values in $M_{CR}$, $M_F$ to 0.5 and $k = 1$;
    // Main loop
4   while The termination criteria are not met do
5       $S_{CR} = \varnothing, S_F = \varnothing$;
6       for $i = 1$ to $N$ do
7           $r_i$ = Select from $[1, H]$ randomly;
8           If $M_{CR,r_i} = \perp$, $CR_{i,G} = 0$. Otherwise
            $CR_{i,G} = \mathrm{randn}_i(M_{CR,r_i}, 0.1)$;
9           $F_{i,G} = \mathrm{randc}_i(M_{F,r_i}, 0.1)$;
10          Generate trial vector $\boldsymbol{u}_{i,G}$ according to
            current-to-$p$best/1/bin;
11      for $i = 1$ to $N$ do
12          if $f(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G})$ then
13              $\boldsymbol{x}_{i,G} \to \boldsymbol{A}$, $CR_{i,G} \to S_{CR}$, $F_{i,G} \to S_F$;
14              $\boldsymbol{x}_{i,G+1} = \boldsymbol{u}_{i,G}$;
15          else
16              $\boldsymbol{x}_{i,G+1} = \boldsymbol{x}_{i,G}$;
17      If necessary, delete randomly selected individuals from the
        archive such that the archive size is $|\boldsymbol{A}|$.
18      Update memories $M_{CR}$ and $M_F$ (Algorithm 2);
19      $G = G + 1$;
```

Algorithm 1: SHADE algorithm

Using the CEC2014 benchmarks [3] as the training data, we apply the algorithm configuration tool SMAC [18] in order to tune the parameters for standard DE, as well as several variants of SHADE [16], [17], a state-of-the-art adaptive DE algorithm, under 3 different scenarios: (1) expensive scenario: $10^2 \times D$ fitness evaluations, (2) medium scenario: $10^4 \times D$ evaluations, and (3) cheap scenario: $10^5 \times D$ evaluations, where D is the benchmark problem dimensionality. The tuned configurations are then tested using the 24 problems in the BBOB noiseless benchmarks [19], [20]. In addition, we compare these tuned DEs to three restart CMA-ES variants that have been shown to perform well on the BBOB benchmarks: HCMA [21], BIPOP-CMA-ES [22], and IPOP-CMA-ES [8].

II. Success-History based Adaptive DE (SHADE)

In this section, we describe SHADE, which is currently one of the state-of-the-art adaptive DE algorithms [16], [17].[1] After first describing SHADE, we describe Restart SHADE (R-SHADE) in Section II-B, L-SHADE in Section II-C, and Restart L-SHADE (RL-SHADE) in Section II-D.

A. SHADE

This section briefly describes SHADE [16], [17], shown in Algorithm 1[2]. 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. $\boldsymbol{A}$ is an external archive [15] which stores inferior individuals. A historical memory M_{CR}, M_F stores a set of CR, F values that have performed well in the past. New CR, F pairs are generated by directly sampling the parameter space close to one of these stored pairs. An index k $(1 \leq k \leq H)$ determines the position in the memory to update.

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

Algorithm 2: Memory update algorithm

After the initialization procedure, trial vector generation and selection are repeated until some termination criterion is encountered. In lines 7–9, in each generation G, the control parameters CR_i and F_i used by each individual $\boldsymbol{x}_i$ are generated by randomly selecting an index r_i from $[1, H]$. In line 8, if M_{CR,r_i} has been assigned the "terminal value" $\perp$, CR_i is set to 0.

In line 10, a mutant vector $\boldsymbol{v}_{i,G}$ is generated from an existing population members by applying the current-to-$p$best/1 mutation strategy: $\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + F_i \cdot (\boldsymbol{x}_{pbest,G} - \boldsymbol{x}_{i,G}) + F_i \cdot (\boldsymbol{x}_{r1,G} - \boldsymbol{x}_{r2,G})$ [15]. Individual $\boldsymbol{x}_{pbest,G}$ is randomly selected from the top $N \times p$ $(p \in [0, 1])$ members in generation G. The individuals $\boldsymbol{x}_{r_1}$ and $\boldsymbol{x}_{r_2}$ are randomly selected from $\boldsymbol{P}$ and $\boldsymbol{P} \cup \boldsymbol{A}$ such that they differ from each other as well as $\boldsymbol{x}_i$.

After generating the mutant vector $\boldsymbol{v}_{i,G}$, it is crossed with the parent $\boldsymbol{x}_{i,G}$ in order to generate trial vector $\boldsymbol{u}_{i,G}$. In SHADE, Binomial Crossover, which is the most commonly used crossover operator in DE, is used and implemented as follows: For each j $(j = 1, ..., D)$, if $\mathrm{rand}[0, 1) \leq CR$ or $j = j_{rand}$, $u_{j,i,t} = v_{j,i,t}$. Otherwise, $u_{j,i,t} = x_{j,i,t}$. $\mathrm{rand}[0, 1)$ denotes a uniformly selected random number from $[0, 1)$, and j_{rand} is a decision variable index which is uniformly randomly selected from $[1, D]$.

In line 11–16, after all of the trial vectors have been generated, a selection process determines the survivors for the next generation. Parent vectors $\boldsymbol{x}_{i,G}$ which were worse than the trial vectors $\boldsymbol{u}_{i,G}$ are preserved in $\boldsymbol{A}$ as line 17.

In each generation, 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 in line 13. Then, at the end of the generation, the memory contents are updated using Algorithm 2. In Algorithm 2, the weighted Lehmer mean $\mathrm{mean}_{WL}(S)$ is computed using the formula below. Where the amount of fitness improvement $\Delta f_k = |f(\boldsymbol{u}_{k,G}) - f(\boldsymbol{x}_{k,G})|$ is used in order to influence the parameter adaptation (S refers to either S_{CR} or S_F).

$$\mathrm{mean}_{WL}(S) = \frac{\sum_{k=1}^{|S|} w_k \cdot S_k^2}{\sum_{k=1}^{|S|} w_k \cdot S_k}, w_k = \frac{\Delta f_k}{\sum_{l=1}^{|S_{CR}|} \Delta f_l} \qquad (1)$$

As M_{CR} is updated, if $M_{CR,k,G} = \perp$ (where $\perp$ denotes a special, "terminal value") or $\max(S_{CR}) = 0$ (i.e., all elements of S_{CR} are 0), $M_{CR,k,G+1}$ is set to $\perp$. Thus, if M_{CR} is

[1] Due to space limitations, we omit a description of standard DE – see [11] for details.

[2] For more details, see [17].

assigned the terminal value $\perp$, then M_{CR} will remain fixed at $\perp$ until the end of the search.

B. R-SHADE: SHADE with Restart

This section describes R-SHADE, which incorporates restarts into SHADE. Restart strategies that reset and restart the search when search progress has stalled have been widely used in the EC community (c.f. [23]). When implementing a restart strategy, the major design decision is the *restart criterion*, which determines when a restart is necessary. If the restart criterion is too aggressive, then the algorithm might restart even though search has not really converged. On the other hand, if the restart criterion is too conservative, then valuable time may be wasted on an unproductive search effort. Many restart criteria have been proposed in the literature. In this paper, we adopt a restart strategy that uses the following 3 criteria (the first two criteria were described in [24]).[3]

(1) Solution vector $\boldsymbol{x}$ convergence

When there exists some $j = (1, ..., D)$ for which Δx_j (defined below) is small, restart because the solution vectors have probably converged. In this paper, ε_x = 1e−12.

$$\exists j \; \Delta x_j < \varepsilon_x \max_{i=1,\ldots,N} \{|x_{i,j}|\} \tag{2}$$

$$\Delta x_j = \max_{i=1,\ldots,N} \{x_{i,j}\} - \min_{i=1,\ldots,N} \{x_{i,j}\} \tag{3}$$

(2) Fitness value $f(\boldsymbol{x})$ convergence

When there exists some $k = (1, ..., N)$ for which Δf_k (defined below) is small, restart because the fitness values have probably converged. In this paper, ε_f = 1e−12.

$$\exists k \; \Delta f_k < \varepsilon_f \max_{i=1,\ldots,N} \{|f_i|\} \tag{4}$$

$$\Delta f_k = \max_{i=1,\ldots,N} \{f_i\} - \min_{i=1,\ldots,N} \{f_i\} \tag{5}$$

(3) Lack of updates to best-so-far solution

If, within a particular restart iteration, the best-so-far solution in its iteration has not been updated for Evals^{stop} steps, then restart because the search has probably stopped making progress. In this paper, we used Evals^{stop} = $500 \times D$.

R-SHADE is a simple modification of SHADE which applies the three restarts describe above. In Algorithm 1, if *any* of restart criteria (1), (2), (3) are met, then the search is restarted starting at line 1.

C. L-SHADE: SHADE with Linear Population Size Reduction Strategy

L-SHADE [17] is a variant of SHADE algorithm with Linear Population Size Reduction (LPSR), a simple deterministic population resizing method and a special case of SVPS [25] which reduces the population linearly, and requires only 1 parameter which needs to be tuned (initial population size N^{init}). LPSR continuously reduces the population to match a linear function where the population size at generation 1 is N^{init}, and the population at the end of the run is N^{min}. At the end of each generation G (line 19), the population size in the next generation, N_{G+1}, is computed according to the formula:

$$N_{G+1} = \text{round}\left[\left(\frac{N^{min} - N^{init}}{\text{MaxEvals}}\right) \cdot \text{Evals} + N^{init}\right] \tag{6}$$

N^{min} is set to the smallest possible value such that the evolutionary operators can be applied. In the case of L-SHADE, $N^{min} = 4$ because the current-to-$p$best mutation operator requires 4 individuals. Evals is the current number of fitness evaluations, and MaxEvals is the maximum number of fitness evaluations.

```
// Initialization phase
MaxEvals$^{1iter}$ = MaxEvals$/B$;
Evals= 0;
// Main loop
while The termination criteria are not met do
    Run L-SHADE with budget MaxEvals$^{1iter}$;
    Evals+ = MaxEvals$^{1iter}$;
    // Adjust MaxEvals$^{1iter}$ to remained budget
    if MaxEvals$^{1iter}$ > MaxEvals − Evals then
        MaxEvals$^{1iter}$ = MaxEvals − Evals;
```

Algorithm 3: RL-SHADE algorithm

For explorative search in the beginning of the search, L-SHADE uses a relatively large initial population size N^{init} and reduces its size gradually. As a result, the search is executed with a small population size and becomes more exploitative. This deterministic population resizing mechanism makes L-SHADE more robust and effective.

D. RL-SHADE: L-SHADE with Restart

As with SHADE, it seems natural to extend L-SHADE by implementing a restart strategy. However, in preliminary experiments, we observed that L-SHADE does not tend to converge until the population size has shrunk to the minimal population size N^{min}, i.e., until MaxEvals evaluations have been performed. This is because L-SHADE starts with a relatively large population size. Thus, instead of the same restart criterion as R-SHADE, RL-SHADE implements a slightly modified restart strategy.

RL-SHADE (Algorithm 3), which is L-SHADE extended with restarts, relies on a single restart criterion which is a modified version of criterion (3) above. Each iteration executes for at least MaxEvals^{1iter} = MaxEvals$/B$ fitness evaluations, where $B \geq 1$. If at least MaxEvals^{1iter} evaluations have been performed and the best-so-far solution has not been updated (improved) within the past $500 \times D$ fitness evaluations, RL-SHADE restarts from scratch. The basic idea is to allocate at least MaxEvals^{1iter} evaluations to each iteration, and then after that point, restart if no recent progress has been made.

[3] In preliminary experiments, there were several multimodal benchmark problems where criteria (1) and (2) failed to detect that search had clearly stalled in some circumstances, so we added (3); we also tested (3) by itself, and found that applying all three criteria performed slightly better than (3) by itself, so we used a combination of all 3 criteria in this study.

TABLE I: For each DE variant, the default control parameter values, as well as the best parameters found by tuning the algorithm with SMAC using CEC2014 benchmark problems $F_1 \sim F_{16}$ (for $D = 2, 10, 20$) as training problems.

(a) R-DE

Parameters	Range	Default	MaxEvals $10^2 \times D$	MaxEvals $10^4 \times D$	MaxEvals $10^5 \times D$
population rate	[0, 10]	5.0	0.15	1.16	1.45
F	[0.1, 1]	0.5	0.74	0.53	0.61
CR	[0, 1]	0.5	0.39	0.31	0.17
strategy	see the text	rand/1	current-to-$p$best/1	best/1	best/1
p	[0, 0.2]	0.05	0.03	n/a	n/a
archive rate	[0, 2]	1.0	0.68	n/a	n/a

(b) R-SHADE

Parameters	Range	Default	MaxEvals $10^2 \times D$	MaxEvals $10^4 \times D$	MaxEvals $10^5 \times D$
population rate	[0, 10]	5.0	0.45	3.74	3.96
initial M_F	[0, 1]	0.5	0.90	0.53	0.38
initial M_{CR}	[0, 1]	0.5	0.06	0.71	0.94
p	[0, 0.2]	0.05	0.01	0.13	0.09
archive rate	[0, 2]	1.0	1.92	0.65	0.12
memory size	[1, 20]	10	16	10	11

(c) RL-SHADE

Parameters	Range	Default	MaxEvals $10^2 \times D$	MaxEvals $10^4 \times D$	MaxEvals $10^5 \times D$
initial population rate	[10, 20]	15	5.19	13.63	16.39
initial M_F	[0, 1]	0.5	0.84	0.93	0.28
initial M_{CR}	[0.1, 1]	0.5	0.12	0.72	0.43
p	[0, 0.2]	0.05	0.01	0.09	0.02
archive rate	[0, 2]	1.0	1.13	1.86	0.94
memory size	[1, 20]	10	7	3	7
B	[1, 10]	1	8	1	5

Although we omit the data due to space constraints, RL-SHADE outperforms standard L-SHADE for cheap scenarios (MaxEvals = $10^5 \times D$).

III. Tuning the Parameters Using SMAC

A. Settings

In this section, we describe parameter tuning of R-DE, R-SHADE, and L-SHADE using the automated algorithm configuration tool, SMAC. Where, R-DE is the standard DE algorithm [11] with restart strategy as same with R-SHADE described in Section II-B.

In recent years, automated algorithm configuration has been an active area of research in both the AI and EC communities, and within the DE community, our previous work on L-SHADE has demonstrated the utility of an algorithm configuration tool for parameter tuning [17]. An algorithm configurator takes as input an algorithm executable, a formal description of the parameters for the algorithm, and a set of training problem instances. It searches the space of possible parameter values by repeatedly generating a candidate configuration (e.g., by local search) and evaluating the configuration on the set of the training instances (or some intelligently selected subset of training instances). The configuration with highest expected utility on the training set is returned. Well-known algorithm configurators include ParamILS [26], irace [27], and SMAC [18]. In this paper, we use SMAC, which is a surrogate-model based configurator which can be used to tune real-valued, integer-valued, categorical, and conditional parameters [18]. We used the most recent version of SMAC downloaded from the authors' website[4].

The evaluation function used by SMAC to assess the quality of a candidate DE configuration was the mean of the difference between the solution found by the DE configuration and the optimal value for each benchmark function in the training set, consisting of functions $F_1 \sim F_{16}$ in 2, 10, and 20 dimensions (i.e., $16 \times 3 = 48$ problems) from the CEC2014 benchmarks [3][5]. We generated sets of tuned parameters for 3 different training scenarios (the DE algorithms were tuned for 3 different fitness evaluation limits): (1) expensive scenario – $10^2 \times D$ evaluations, (2) medium scenario – $10^4 \times D$ evaluations, and (3) cheap scenario – $10^5 \times D$ evaluations. Each run of SMAC was limited to $3,000$ DE configurations. For each DE variant, for each training scenario, SMAC was run 5 times, and we selected the best result out of these 5 runs. Finally, SMAC itself has some parameters that control the algorithm configurator; we used the default parameters for these.

B. SMAC Results

For each DE variant, the default values of the control parameters (from [15], [16], [17]), the ranges for the parameters, as well as the values found by SMAC, are shown in Table I.

Binomial crossover was used for all DE variants. For R-DE, 7 possible mutation strategies, rand/1, rand/2, best/1, best/2, current-to-best/1, current-to-best/2, and current-to-$p$best/1 with archive could be selected.[6] The "current-to-$p$best/1 with archive" strategy has control parameter p and archive rate; these are modeled as conditional parameters [18] in SMAC. The population size $N = \max(\mathrm{round}(\text{population rate} \times D), 6)$, and archive size $|\boldsymbol{A}| = \mathrm{round}(\text{archive rate} \times N)$. Note that the population size was set to be at least 6 in order to be compatible with the rand/2 mutation strategy, which requires at least 6 individuals.

In Table I(a) and (b), when MaxEvals = $10^2 \times D$, the tuned population rates are 0.15 and 0.45, respectively, which result in population sizes (recall from above that population size = max(round(population rate × D), 6)) which are significantly lower than the standard population size of 100 suggested in the literature [15], [16]. This is because with a very small number of fitness evaluations (MaxEvals = $10^2 \times D$), a small

[4] http://www.cs.ubc.ca/labs/beta/Projects/SMAC/

[5] We excluded $F_{17} \sim F_{30}$ because 2-dimensional versions were not included in [3].

[6] For details on these mutation strategies, see the survey by [12].

population size is selected so that it is possible to greedily search a focused area of the search space. On the other hand, as MaxEvals increased from $10^2 \times D$ to $10^4 \times D$ to $10^5 \times D$, the population rate also increases, suggesting that as MaxEvals increases, a less focused search that performs more exploration leads to better performance. This tendency can be seen in the tuned population rate of RL-SHADE.

As shown in Table I(c), when RL-SHADE is run with MaxEvals = $10^2 \times D$ and $10^5 \times D$, the restart frequency parameter B is set to 8, 5 (restart frequently, approximately 8-5 times during the run). However, when MaxEvals= $10^4 \times D$, B = 1, i.e., no restarts (same as plain L-SHADE). Thus, the behavior of RL-SHADE changes dramatically depending on MaxEvals.

As shown above, the results of parameter tuning vary significantly depending on MaxEvals. This is consistent with previous results for ACO [7] and IPOP-CMA-ES [9], [10]. Thus, in practice, it is vital to carefully consider the available computational budget when tuning DE algorithms.

IV. Results

In this section, we evaluate the tuned parameter settings for R-DE, R-SHADE, RL-SHADE obtained in Section III using SMAC. For the test problems, we use the 24 problems in the BBOB noiseless benchmark set [19], [20] (note that these differ from the CEC2014 benchmarks [3] used as the training problems).

We compare the DE variants to three CMA-ES variants that are known well to perform well on the BBOB benchmarks, HCMA [21], BIPOP-CMA-ES [22] and IPOP-CMA-ES [8]. IPOP-CMA-ES, upon which HCMA and BIPOP-CMA-ES are based, incorporates a restart strategy into the basic CMA-ES algorithm [28], and doubles the population size after each restart, broadening the search after each restart. For IPOP-CMA-ES, we used the data for IPOP-CMA-ES-tany and IPOP-CMA-ES-texp, which are results using control parameters tuned for anytime, expensive scenarios which was provided in [9]. BIPOP-CMA-ES first executes CMA-ES with a default population size. Then, it divides the remainder of the time available evenly between IPOP-CMA-ES and multistart CMA-ES with a small population size. HCMA [21] is a hybrid method which incorporates a surrogate model and two local search methods (NEWOUA [29] and STEP [30]) into BIPOP-CMA-ES. All experimental data were downloaded from the BBOB website [31].

A. Impact of Budget Scenario on Tuned Parameters

In this section, we evaluate the parameter settings obtained for various MaxEvals in Section III. Figure 1 shows the Empirical Cumulative Distribution Function (ECDF) for each algorithm, each parameter setting, for 24 BBOB benchmark problems (10 dimensions) for MaxEvals = $\{10^2 \times D, 10^4 \times D, 10^5 \times D\}$. After the DE variant name, 10e2 indicates the results for tuning with MaxEvals= $10^2 \times D$, 10e4 is for MaxEvals= $10^4 \times D$, and 10e5 is for MaxEvals= $10^5 \times D$.

The results for R-DE in Figure 1(a) show that for MaxEvals = $10^2 \times D$, R-DE-10e4 performs slightly better than R-DE-10e2, and R-DE-10e5 is clearly worse than R-DE-10e2. However, for MaxEvals = $10^4 \times D$, R-DE-10e5 had the best performance, and for MaxEvals = $10^5 \times D$, R-DE-10e5 and R-DE-10e4 perform similarly. In contrast, for MaxEvals = $10^4 \times D$ and $10^5 \times D$, R-DE-10e2 performs poorly.

A similar trend can be seen for R-SHADE and RL-SHADE. Figure 1(b) shows that although R-SHADE-10e2 performs well for MaxEvals = $10^2 \times D$, it performs worse than R-SHADE-10e4 and R-SHADE-10e5 for MaxEvals = $10^4 \times D$ and MaxEvals= $10^5 \times D$. Figure 1(c) shows that RL-SHADE-10e2, RL-SHADE-10e4 and RL-SHADE-10e5 performs best for MaxEvals = $10^2 \times D, 10^4 \times D$, and $10^5 \times D$ respectively.[7] However, when the computational budget for the training phase and the testing phase are different, RL-SHADE tends to perform poorly.

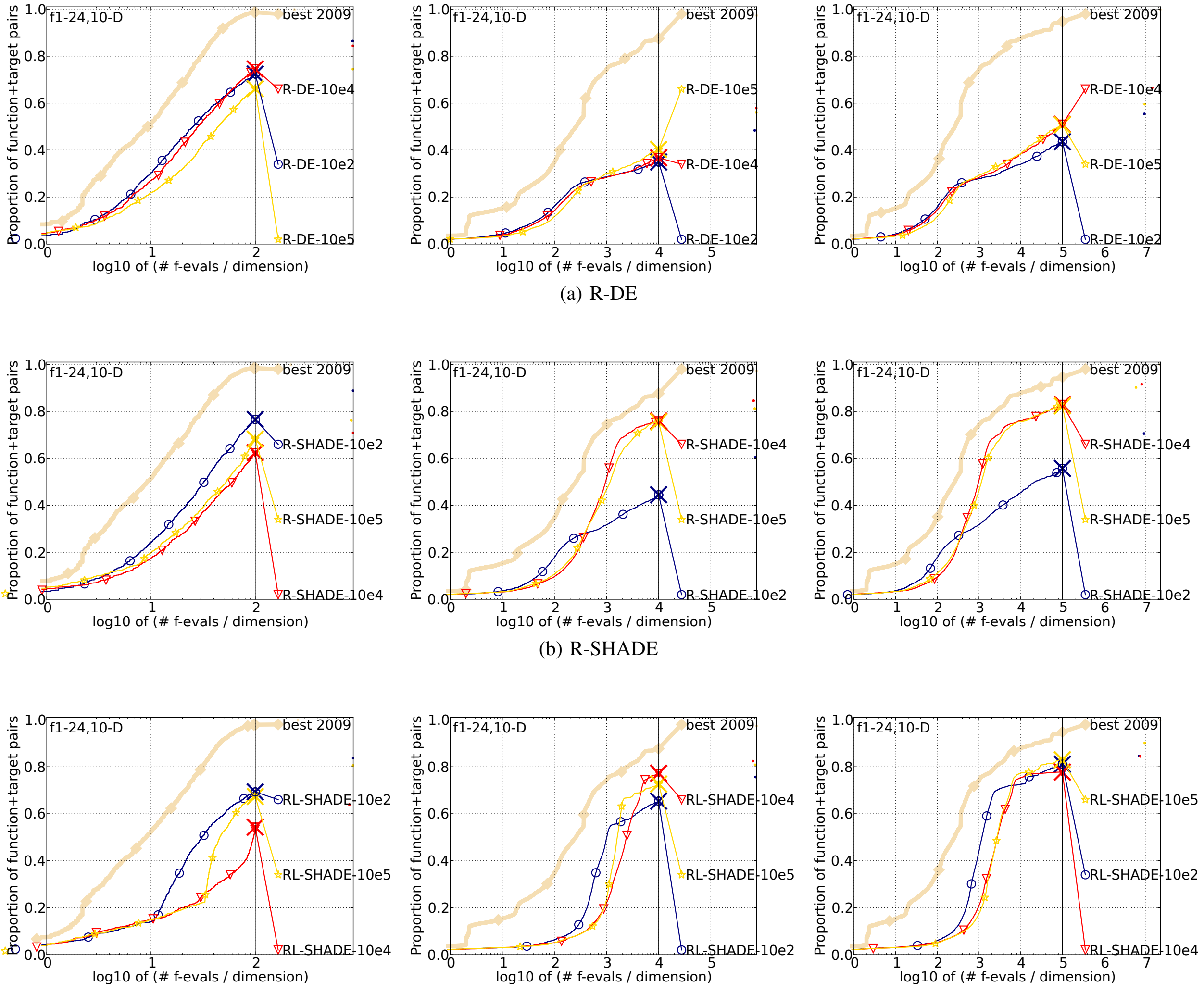
In summary, it appears that the computational budget (MaxEvals) used during parameter tuning has a significant impact on performance of R-DE, R-SHADE, and RL-SHADE when tested under different MaxEvals settings. Thus, when MaxEvals used for training (tuning) and testing differ significantly, it is likely that the parameters obtained by tuning are inappropriate for the test problem, and one can not expect good parameters when using parameters optimized for a computational budget that significantly differ from the target application scenario. If MaxEvals for a particular application scenario of DE is known a priori, then it appears that tuning the control parameters using a computational budget similar to MaxEvals is necessary in order to maximize performance.

B. Comparing DE algorithms with state-of-the-art restart CMA-ES variants

In this section, we compare DE algorithms to state-of-the-art restart CMA-ES variants (HCMA, BIPOP-CMA-ES, IPOP-CMA-ES). Figures 2 and 3 show the results for expensive (MaxEvals = $10^2 \times D$) and cheap (MaxEvals = $10^5 \times D$) scenarios for $D = 2, 3, 5, 10, 20$-dimensional BBOB benchmarks (all 24 problems). For R-DE, R-SHADE, and RL-SHADE, we tuned the parameters separately for expensive and cheap scenarios. Since IPOP-CMA-ES-texp was designed and tuned for expensive scenarios and BIPOP-CMA-ES was tuned/designed for cheap scenarios, for fairness, we only include IPOP-CMA-ES-texp data for the expensive scenario, and BIPOP-CMA-ES data for the cheap scenario.

Figure 2 shows that in the expensive scenario, HCMA performs best for all dimensions. For D = 2, 3, and 5 dimensions, R-DE-10e2 outperforms R-SHADE-10e2, RL-SHADE-10e2, IPOP-CMA-ES-tany and IPOP-CMA-ES-texp. In addition, R-DE-10e2 performs better than RL-SHADE-10e2 for all dimensions. This contradicts the widely held belief that

[7] Although the restart frequency parameter B for RL-SHADE-10e2 and RL-SHADE-10e5 is set to 8 and 5 respectively (see Table I(c)), we observed that they never restarted for MaxEvals = $10^2 \times D$. This is because RL-SHADE continues to run until the best-so-far solution has not been updated for $500 \times D$ evaluations (see Section II-D).



(a) R-DE

(b) R-SHADE

(c) RL-SHADE

Fig. 1: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{[-8..2]}$ for 10 dimensional all functions . The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single target. After the DE variant name, 10e2 indicates the results for tuning with MaxEvals= $10^2 \times D$, 10e4 is for MaxEvals= $10^4 \times D$, and 10e5 is for MaxEvals= $10^5 \times D$.

adaptive DE outperforms standard DE, and shows that methods that are suited for cheap scenarios are not necessarily suited for expensive scenarios. In addition, this result also shows that for low-dimensional, expensive scenarios, it is quite possible for the simple R-DE algorithm to outperform the far more complex methods such as IPOP-CMA-ES.

Figure 3, which shows the results for the cheap scenario, shows that in contrast to the expensive scenario, R-DE-10e5 (which performed relatively well in the expensive setting) performs poorly. Although RL-SHADE-10e5 and R-SHADE-10e5 lose to HCMA for D =5, 10, 20 and to BIPOP-CMA-ES and IPOP-CMA-ES-tany for D =10 and 20, RL-SHADE-10e5 and R-SHADE-10e5 tend to perform quite well for other values of D. Also, while RL-SHADE-10e5 does not perform as well as BIPOP-CMA-ES and IPOP-CMA-ES-tany for $D = 10, 20$ when the number of evaluations is $10^5 \times D$, RL-SHADE-10e5 outperforms both BIPOP-CMA-ES and IPOP-CMA-ES-tany when the number of evaluations is around $= 10^4 \times D$. From this, it seems that for low-dimensional and medium scenarios (MaxEvals= $10^4 \times D$), RL-SHADE is competitive with state-of-the-art restart CMA-ES variants.

However, note that these comparison results on the BBOB benchmarks for cheap scenarios for SHADE variants and restart CMA-ES variants somewhat contradict our previous work [17]. In [17], we showed that on the CEC2014 benchmarks [3], L-SHADE, which is the non-restarting of RL-SHADE outperformed two state-of-the-art restart CMA-ES variants (NBIPOP-$_{\mathrm{A}}$CMA-ES [32] and iCMA-ILS [33]). Previous work by Liao et al [34] has shown that the comparative performance of methods can vary significantly depending

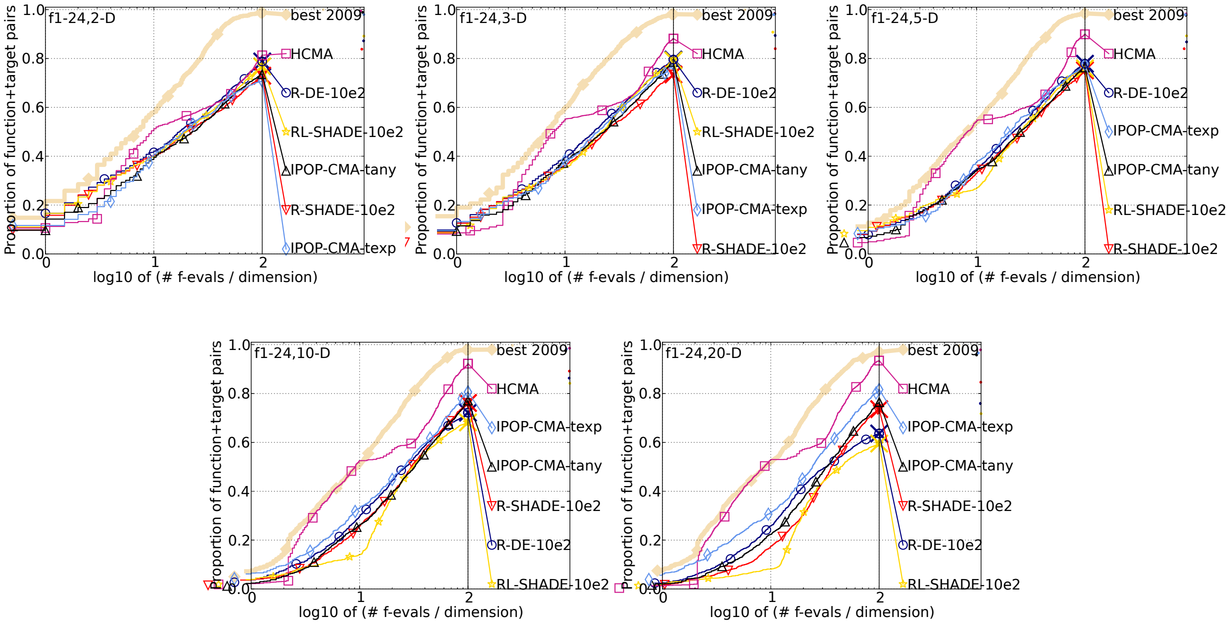


Fig. 2: Comparison of DE algorithms (R-DE, R-SHADE, RL-SHADE) with state-of-the-art restart CMA-ES variants (HCMA, BIPOP-CMA-ES, IPOP-CMA-ES) on BBOB benchmarks for an expensive scenario (MaxEvals = $10^2 \times D$).

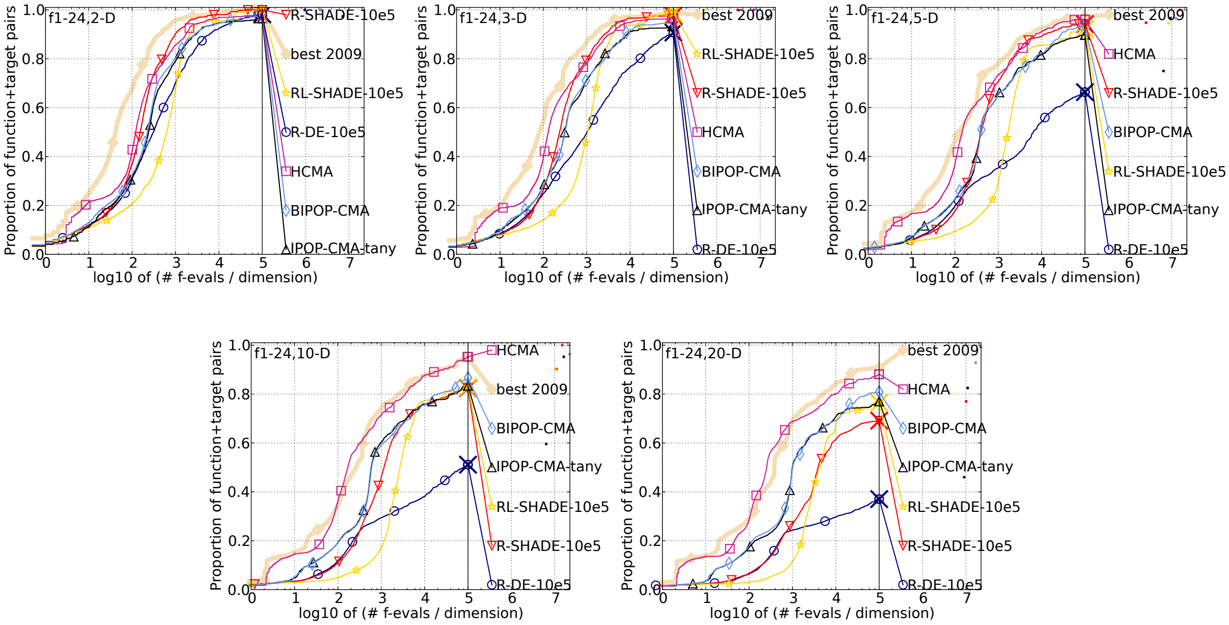


Fig. 3: Comparison of DE algorithms (R-DE, R-SHADE, RL-SHADE) with state-of-the-art restart CMA-ES variants (HCMA, BIPOP-CMA-ES, IPOP-CMA-ES) on BBOB benchmarks for a cheap scenario (MaxEvals = $10^5 \times D$).

on the benchmarks used. In a comparison of 7 methods including IPOP-CMA-ES using the SOCO benchmarks [4] and CEC2005 benchmarks [2], and showed that the rank ordering of the methods varied significantly depending on the benchmarks. An in-depth study of the effect of benchmark problem set selection on the relative ranking of DE and restart CMA-ES algorithms is an avenue for future work.

V. Conclusion

We presented a parameter tuning study of restarting variants of standard DE as well as the state-of-the-art adaptive DE, SHADE. Using the SMAC automated algorithm configuration tool and the CEC2014 benchmarks as training problems, we tuned R-DE, R-SHADE, and RL-SHADE for three scenarios: (1) expensive scenario: MaxEvals= $10^2 \times D$, (2) medium scenario: MaxEvals= $10^4 \times D$, (3) cheap scenario: MaxEvals= $10^5 \times D$. We found that the parameter settings found by SMAC depend significantly on MaxEvals. Each of the tuned parameter settings were then tested on the BBOB noiseless benchmarks under all three scenarios (expensive/medium/cheap). We showed that when MaxEvals is the same for both tuning and testing, good performance can be expected, but performance can be poor if MaxEvals for tuning and testing are not the same.

The tuned DEs were compared with state-of-the-art restart CMA-ES variants on the expensive and cheap scenarios. For $D = 2, 3, 5$ dimensions in the expensive scenario, the simple, restarting standard DE (R-DE) had the best performance among all DE variants, and was competitive with the restart CMA-ES variants, excluding HCMA (which includes a surrogate-based component specialized for expensive scenarios). In the case of cheap scenarios, for low-dimensional problems, R-SHADE and RL-SHADE were competitive with restart CMA-ES variants, and for higher dimensions ($D = 10, 20$), RL-SHADE outperforms BIPOP-CMA-ES and IPOP-CMA-ES when the number of evaluations is around $10^4 \times D$.

Our study showed that with tuning, a simple, restarting version of standard DE can be surprisingly effective for low-dimensional problems in an expensive optimization setting. On the other hand, the more sophisticated restarting SHADE variants perform well for medium and expensive settings, and are competitive with restart CMA-ES variants depending on the number of evaluations and the dimensionality. However, in an expensive scenario, SHADE variants are not competitive with HCMA. These results suggest that integration of a surrogate-based component into SHADE as an interesting line of future work which could result in a competitive adaptive DE algorithm for expensive optimization.

Acknowledgment

This work was supported by JSPS KAKENHI Grants 269528, 2324002 and 25330253.

References

[1] X. Yao, Y. Liu, and G. Lin, "Evolutionary Programming Made Faster," *IEEE Tran. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, 1999.

[2] 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 Univ., Tech. Rep., 2005.

[3] 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.

[4] F. Herrera, M. Lozano, and D. Molina, "Test suite for the spec. iss. of Soft Computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems," Univ. of Granada, Tech. Rep., 2010.

[5] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Comput.*, vol. 9, no. 1, pp. 3–12, 2005.

[6] F. G. Lobo, C. F. Lima, and Z. Michalewicz, Eds., *Parameter setting in evolutionary algorithms*, ser. Studies in Computational Intelligence. Springer, 2007.

[7] L. P. Cáceres, M. López-Ibáñez, and T. Stützle, "Ant colony optimization on a budget of 1000," in *ANTS*, 2014, pp. 50–61.

[8] A. Auger and N. Hansen, "A restart cma evolution strategy with increasing population size," in *IEEE CEC*, 2005, pp. 1769–1776.

[9] T. Liao and T. Stützle, "Expensive optimization scenario: IPOP-CMA-ES with a population bound mechanism for noiseless function testbed," in *GECCO (Companion)*, 2013, pp. 1185–1192.

[10] T. Liao, M. A. M. de Oca, and T. Stützle, "Computational results for an automatically tuned CMA-ES with increasing population size on the cec'05 benchmark set," *Soft Comput.*, vol. 17, no. 6, pp. 1031–1046, 2013.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] ——, "Improving the Search Performance of SHADE Using Linear Population Size Reduction," in *IEEE CEC*, 2014, pp. 1658–1665.

[18] F. Hutter, F. H. Hoos, and K. Leyton-Brown, "Sequential Model-Based Optimization for General Algorithm Configuration," in *LION*, 2011, pp. 507–523.

[19] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking 2012: Experimental setup," INRIA, Tech. Rep., 2012.

[20] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Tech. Rep. RR-6829, 2009, updated February 2010.

[21] I. Loshchilov, M. Schoenauer, and M. Sebag, "Bi-population CMA-ES agorithms with surrogate models and line searches," in *GECCO (Companion)*, 2013, pp. 1177–1184.

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

[23] A. S. Fukunaga, "Restart scheduling for genetic algorithms," in *PPSN*, 1998, pp. 357–366.

[24] M. Zhabitsky and E. Zhabitskaya, "Asynchronous Differential Evolution with Adaptive Correlation Matrix," in *GECCO*, 2013, pp. 455–462.

[25] J. L. J. Laredo, C. Fernandes, J. J. M. Guervós, and C. Gagné, "Improving genetic algorithms performance via deterministic population shrinkage," in *GECCO*, 2009, pp. 819–826.

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

[27] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace Package: Iterated Racing for Automatic Algorithm Configuration," IRIDIA, Université Libre de Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2011-004, 2011.

[28] N. Hansen and S. Kern, "Evaluating the CMA Evolution Strategy on Multimodal Test Functions," in *PPSN*, 2004, pp. 282–291.

[29] M. Powell, "The NEWUOA software for unconstrained optimization without derivatives," in *Large-scale nonlinear optimization*, 2006, pp. 255–297.

[30] S. Swarzberg, G. Seront, and H. Bersini, "STEP: The easiest way to optimize a function," in *IEEE CEC*, 1994, pp. 519–524.

[31] COCO, http://coco.gforge.inria.fr/doku.php, 2009.

[32] I. Loshchilov, "Cma-es with restarts for solving cec 2013 benchmark problems," in *IEEE CEC*, 2013, pp. 369–376.

[33] T. Liao and T. Stützle, "Benchmark results for a simple hybrid algorithm on the cec 2013 benchmark set for real-parameter optimization," in *IEEE CEC*, 2013, pp. 1938–1944.

[34] T. Liao, D. Molina, and T. Stützle, "Performance evaluation of automatically tuned continuous optimizers on different benchmark sets," *Applied Soft Comput.*, 2015 (in press).