
Book Section:
Sharma, Mudita, López-Ibáñez, Manuel and Kazakov, Dimitar Lubomirov orcid.org/0000-0002-0637-8106 (2018) Performance Assessment of Recursive Probability Matching for Adaptive Operator Selection in Differential Evolution. In: 15th Intl Conf. on Parallel Problem Solving from Nature. LNCS. Springer, pp. 321-333.

https://doi.org/10.1007/978-3-319-99259-4_26

Performance Assessment of Recursive Probability Matching for Adaptive Operator Selection in Differential Evolution

Mudita Sharma[1], Manuel López-Ibáñez[2], and Dimitar Kazakov[1]

[1] University of York, UK
{ms1938,dimitar.kazakov}@york.ac.uk
[2] University of Manchester, UK
manuel.lopez-ibanez@manchester.ac.uk

Abstract. Probability Matching is one of the most successful methods for adaptive operator selection (AOS), that is, online parameter control, in evolutionary algorithms. In this paper, we propose a variant of Probability Matching, called *Recursive Probability Matching* (*RecPM-AOS*), that estimates reward based on progress in past generations and estimates quality based on expected quality of possible selection of operators in the past. We apply *RecPM-AOS* to the online selection of mutation strategies in differential evolution (DE) on the BBOB benchmark functions. The new method is compared with two AOS methods, namely, *PM-AdapSS*, which utilises probability matching with relative fitness improvement, and *F-AUC*, which combines the concept of area under the curve with a multi-arm bandit algorithm. Experimental results show that the new tuned *RecPM-AOS* method is the most effective at identifying the best mutation strategy to be used by DE in solving most functions in BBOB among the AOS methods.

Keywords: Parameter control· probability matching· differential evolution· black-box optimisation

1 Introduction

In many optimisation algorithms, there are operations, such as crossover, mutation, and neighbourhood exploration, for which a discrete number of operators or strategies exist. Choosing the right operator is often key for improving the performance of the algorithm. Adaptive Operator Selection (AOS) is a framework that dynamically selects an operator at run-time from a finite set of choices. AOS methods are a subset of online tuning or parameter control methods [11]. Examples of AOS methods include *PM-AdapSS* [7] and *F-AUC* [5], both of which were introduced in the context of selecting a mutation strategy in differential evolution (DE) [13]. In particular, *PM-AdapSS* uses probability matching (PM) as the method for operator selector, whereas *F-AUC* uses a method inspired by multi-armed bandits. PM was initially proposed in the context of classifier systems [6] and it was later adapted as a component of AOS methods [7].

In this work, we propose a variant of PM called Recursive Probability Matching (*RecPM*). PM probabilistically selects an operator to apply according to its estimated *quality*. The quality of each operator is calculated as the weighted sum of a reward value, which measures the impact of the most recent application of the operator on solution fitness, and its historical quality. Instead, our proposed *RecPM* estimates the quality of each operator according to a method inspired by the Bellman equation from reinforcement learning [16], which takes into account not only the reward values but also the selection probabilities of other operators. By combining *RecPM* with a credit assignment method based on offspring survival rate, we obtain the *RecPM-AOS* method.

We follow previous works [4], and apply *RecPM-AOS* to adaptively select a mutation strategy in DE for continuous optimisation, and compare our results with both *PM-AdapSS* and *F-AUC*. For completeness, we also include two state-of-the-art algorithms: a variant of DE, *JADE* [17], and an evolution strategy with covariance matrix adaptation, *CMAES* [10]. As a benchmark, we use the noiseless functions from the black-box optimisation benchmark (BBOB) [8]. Our results show that RecPM-AOS is competitive with other AOS methods.

2 Background

2.1 Adaptive Operator Selection

Adaptive Operator Selection (AOS) methods dynamically select, at each iteration t of an algorithm, one operator k out of a discrete set of K operators. The selection is based on (*1*) a credit or *reward* value $r_{k,t}$ that rewards recent performance improvements attributed to the application of the operator and (*2*) an estimated *quality* of the operator $q_{k,t}$ that accumulates historical performance or takes into account the performance of other operators. We identify two components of AOS methods: the credit assignment and the operator selection.

Credit Assignment (CA) defines the performance statistics that measure the impact of the application of an operator and assigns a reward value $r_{k,t}$ according to this impact. For example, the reward of a mutation operator may be defined in terms of the fitness of the solutions generated by its application. CA is applied after each application of the operator, possibly taking into account its past performance. For example, the CA of *F-AUC* uses a sliding window of size W to store the rank-transformed fitness obtained by the last W selected operators that generated an improved solution. A decay factor is applied to the ranks so that top-ranks are rewarded more strongly. The ranks in the window are used to compute a curve of the contribution of each operator and the area under the curve (AUC) is taken as the reward value of the operator. More details are given in the original paper [5]. On the other hand, *PM-AdapSS* only considers the immediate performance of the operators and calculates the reward of selected operator k at iteration t as:

$$r_{k,t} = \frac{1}{N^{\text{surv}}} \sum_{i=1}^{N^{\text{surv}}} \frac{f(\boldsymbol{x}_{\text{best}}) \cdot |f(\boldsymbol{x}_i^{\text{parent}}) - f(\boldsymbol{x}_i)|}{f(\boldsymbol{x}_{\text{i}})} \tag{1}$$

Table 1. Comparison of AOS methods and their components.

		F-AUC	PM-AdapSS	RecPM-AOS
CA		Area Under the Curve	Average Relative Fitness Improvement (Eq. 1)	Offspring Survival Rate (Eq. 10)
OS		Multi-Armed Bandit	Probability Matching (Sec. 2.1)	Probability Matching (Sec. 2.1)
OS	**Selection**	Greedy	Roulette wheel	Roulette wheel
	Quality	UCB (Eq. 2)	Weighted sum of quality and reward (Eq. 3)	Bellman equation (Eq. 9)
Parameters		Window size(W), Scaling Factor(C)	$p_{\min}$, α	$p_{\min}$, γ

where N^{surv} is the number of offspring that improved over its parent, and $f(\boldsymbol{x}_{\text{i}})$, $f(\boldsymbol{x}_i^{\text{parent}})$, and $f(\boldsymbol{x}_{\text{best}})$ are the fitness of an offspring solution generated by selected operator k, of its parent solution and of the best solution found so far, respectively. If there is no improvement or the operator was not selected at iteration t, the reward is zero.

The Operator Selector (OS) estimates the quality $q_{i,t+1}$ of each operator i, based on the reward assigned to it at iteration t, and chooses one operator to use in iteration $t+1$ among K operators according to its quality. For example, the OS in *F-AUC* uses a multi-arm bandit (MAB) technique called Upper Confidence Bound (UCB) [1] to calculate:

$$q_{i,t+1} = r_{i,t} + C \cdot \sqrt{\frac{2 \log \sum_{j=1}^{K} n_{j,t}}{n_{i,t}}} \tag{2}$$

where C is a scaling factor parameter, $n_{i,t}$ is the number of applications of operator i in the last W iterations that improved a solution, and $r_{i,t}$ is the reward assigned to operator i at iteration t. In the above equation, $r_{i,t}$ introduces exploitation whereas the second term introduces exploration. The operator selector greedily chooses the operator with the highest quality value. By comparison, *PM-AdapSS* uses probability matching (PM) to map the quality of each operator to a probability value and applies roulette-wheel selection to probabilistically choose the next operator. In particular, the quality of each operator is calculated as:

$$q_{i,t+1} = q_{i,t} + \alpha \cdot (r_{i,t} - q_{i,t}) \,, \forall i \in K \tag{3}$$

where α is a parameter called adaptation rate. The selection probabilities for choosing an operator in iteration $t+1$ are calculated as:

$$p_{i,t+1} = p_{\min} + (1 - K \cdot p_{\min}) \left(\frac{q_{i,t+1}}{\sum_{j=1}^{K} q_{j,t+1}} \right) \tag{4}$$

where $p_{\min}$ is a minimum probability of selection. Initially, $q_{i,0} = 1$ and $p_i = 1/K$, $\forall i \in K$. Thus initially all operators have the same chance of getting selected.

Table 1 summarizes the components of the various AOS methods compared in this paper. The components of the proposed *RecPM-AOS* are described in the following Section 3.

Algorithm 1 DE with AOS

```
Initialise parameter values of DE (F, NP, CR) and AOS method
Initialise and evaluate fitness of each individual x_i in the population
t = 0 (generation number or time step)
while stopping condition is not satisfied do
  for each x_i, i = 1, ..., NP do
    if one or more operators not yet applied then
      k = Uniform selection among operator(s) not yet applied
    else
      k = Select mutation strategy based on selection method (AOS)
    Generate offspring using selected operator k
  Evaluate offspring population
  Perform credit assignment (AOS)
  Estimate quality for each operator (AOS)
  Update selection value (eg. probability) for each operator (AOS)
  t = t + 1
```

2.2 Mutation strategies in Differential Evolution

In order to evaluate different AOS methods, we apply them to the online selection of mutation strategies in differential evolution (DE) [13]. In DE, the mutation strategy creates an offspring solution $\boldsymbol{u}$ as a linear combination of three or more parent solutions $\boldsymbol{x}_i$, where i is the index of a solution in the current population. Different strategies show a different balance between exploration and exploitation in the search space and they may be applied to the current solution, a random one or the best one. Examples of such mutation strategies [2] are:

$$\begin{aligned}
\text{“DE/rand/1”:} \quad & \boldsymbol{u}_i = \boldsymbol{x}_{r_1} + F \cdot (\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}) \\
\text{“DE/rand/2”:} \quad & \boldsymbol{u}_i = \boldsymbol{x}_{r_1} + F \cdot (\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3} + \boldsymbol{x}_{r_4} - \boldsymbol{x}_{r_5}) \\
\text{“DE/rand-to-best/2”:} \quad & \boldsymbol{u}_i = \boldsymbol{x}_{r_1} + F \cdot (\boldsymbol{x}_{\text{best}} - \boldsymbol{x}_{r_1} + \boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3} + \boldsymbol{x}_{r_4} - \boldsymbol{x}_{r_5}) \\
\text{“DE/current-to-rand/1”:} \quad & \boldsymbol{u}_i = \boldsymbol{x}_i + F \cdot (\boldsymbol{x}_{r_1} - \boldsymbol{x}_i + \boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3})
\end{aligned}$$

where F is a parameter, and r_1, r_2, r_3, r_4, and r_5 are randomly generated indexes.

For a fair comparison, all AOS methods in this paper are integrated into the same DE algorithm and able to select from the set of mutation strategies shown above. The general framework of DE with AOS is shown in Algorithm 1.

3 Recursive PM (RecPM)

We propose here a novel *PM* variant called Recursive Probability Matching (*RecPM*). The main difference between *PM* and *RecPM* is that the latter estimates the quality of each operator by adapting the Bellman equation from Markov Decision Processes (MDPs) [14,16]. MDP is a framework from Reinforcement Learning for making decisions in a stochastic environment. MDP assumes that the current state is independent of the whole history given the previous state (Markov property). Bellman equation [14] is widely used in MDPs to calculate

the expected return starting from a state. Although other AOS and parameter control methods have used techniques from MDP such as Q-learning and SARSA [11], our proposal is the first to be based on Bellman equation, to the best of our knowledge.

In the context of AOS, a state represents the selected operator k at a time step t and the corresponding reward is the future immediate reward assigned to the operator $r'_{k,t+1}$, which is based on the impact of the application of the operator on the performance of the algorithm. Since the next operator is chosen probabilistically, we consider only transitions between states and rewards, and not actions, thus we follow the Bellman equation for discrete decision processes [14, p. 3094], which is used to predict the next state according to the expected next reward given the current state. Our motivation for using the Bellman equation is to use the historical performance of operators to predict their quality in the next iteration, which is then mapped to their probability of selection.

We use the Bellman equation to estimate the quality $q_{k,t}$ of an operator k after its application in iteration t as the expected value ($\mathrm{E}[\cdot]$) of its total sum of discounted future rewards:

$$q_{k,t} = \mathrm{E}[r'_{k,t+1} + \gamma r'_{k,t+2} + \cdots \mid K_t = k] = \mathrm{E}[\sum_{z=0}^{\infty} \gamma^z r'_{k,t+z+1} \mid K_t = k] \tag{5}$$

$$= \mathrm{E}[r'_{k,t+1} + \gamma \sum_{z=0}^{\infty} \gamma^z r'_{k,t+z+2} \mid K_t = k] \qquad \text{(using recursive property)} \tag{6}$$

$$= r_{k,t+1} + \sum_{j=1}^{K} P_{kj} \left[\gamma \, \mathrm{E} \left[\sum_{z=0}^{\infty} \gamma^z r'_{k,t+z+2} \mid K_{t+1} = j \right] \right] \quad \text{(assuming } \mathrm{E}[r'_k] = r_k) \tag{7}$$

$$= r_{k,t+1} + \gamma \sum_{j=1}^{K} P_{kj} q_{j,t+1} \quad \text{(using definition of } q_{k,t} \text{ in Eq. 5)} \tag{8}$$

where $r_{k,t+1}$ is the accumulated reward that stores all the past achievements (accumulated reward) for operator k and γ is the discount rate. In the context of AOS, we do not know the probability matrix P of size $K \times K$, thus we decided to calculate each entry as $P_{k,j} = p_k + p_j$, that is, as the sum of the selection probabilities of operators k and j.

The rationale behind the formula above is as follows: When estimating $q_{k,t}$, operator k competes with all other operators $j \in K$, including itself, since the selection of other operators in the past has impact on the current performance of the selected operator. Thus, their probabilities are added and multiplied by by the quality estimate of operator j. These values are then aggregated in the end to get an overall estimate for operator k. The quality is an estimate not because of the expected values, which are assumed to be completely provided by the method, but because $q_{j,t+1}$ is not known and the current estimate at t is used instead. When considering all operators, this forms a system of linear

equations and can be re-written in the following vector form:

$$\boldsymbol{Q}_t = \boldsymbol{R}_{t+1} + \gamma P \boldsymbol{Q}_t \quad \text{or} \quad \boldsymbol{Q} = (1 - \gamma P)^{-1} \boldsymbol{R} \tag{9}$$

where $\boldsymbol{Q}$ and $\boldsymbol{R} = [r_i]$ are the K-dimensional quality and reward vectors that are updated at the end of each iteration t. The system of linear equations can be solved efficiently by matrix inversion [14] when the number of operators is small. $\boldsymbol{Q}$ is then normalised using the softmax function, which "squashes" each real value to a K-dimensional vector in the range $(0, 1)$ using the exponential function. Once the quality is estimated for each operator, the probability vector $\boldsymbol{p} = [p_i]$ and probability matrix P are updated as in *PM-AdapSS* (Eq. 4) and used for the selection of an operator.

RecPM utilises the steps of Probability Matching as described in section 2.1 except for the definition of operator quality, which is estimated using the Bellman equation as shown above. However, to obtain an AOS method, we still need to specify the credit assignment method that updates the reward values after the application of the selected operators at time step t. We propose to calculate the immediate reward $r'_{k,t+1}$ assigned to the selected operator as the number N_t^{surv} of offspring that survive to the next generation $t+1$ divided by the population size NP. We define the accumulated reward $r_{k,t}$ assigned to an operator as the ratio of offspring that survived plus half the last accumulated reward. The remainder unselected operators receive half of their accumulated reward. Thus, each operator gets a fraction of last reward value, that stores its historical performance, and the selected one gets extra reward. The value of 0.5 as weight assigned to $r_{k,t}$ was chosen by intuition.

$$r_{i,t+1} = \begin{cases} r'_{k,t+1} + 0.5 \cdot r_{k,t}, & \text{if } k \text{ is selected} \\ 0.5 \cdot r_{i,t}, & \forall i \neq k \end{cases}, \text{ where } r'_{k,t+1} = \frac{N_t^{\text{surv}}}{NP} \tag{10}$$

The rational behind this credit assignment is that, if the operator is unlucky and not getting selected for enough number of generations, it still receives some reward based on its past performance and it has a chance of being selected in the future. This ensures that such operator is not discarded completely and may be selected after a certain number of generations.

The combination of *RecPM* with the above credit assignment leads to a new AOS method named *RecPM-AOS* in the following. When comparing *PM-AdapSS* and *RecPM-AOS*, the former uses *PM* as an operator selector whereas the latter uses *RecPM*. Both AOS methods use a credit assignment based on the number of improvements from parent to offspring (N^{surv}), however, *PM-AdapSS* uses average relative fitness improvement (Eq. 1) as immediate reward without using accumulated reward, whereas *RecPM-AOS* uses offspring survival rate as immediate reward combined with a fraction of its previous accumulated reward.

RecPM-AOS is integrated within DE (Algorithm 1) to make DE more efficient by adaptively selecting, at run-time, a mutation strategy among the four mutation strategies shown in section 2.2. DE combined with *RecPM-AOS* has five parameters: three belong to DE, namely, scaling factor (F), population size

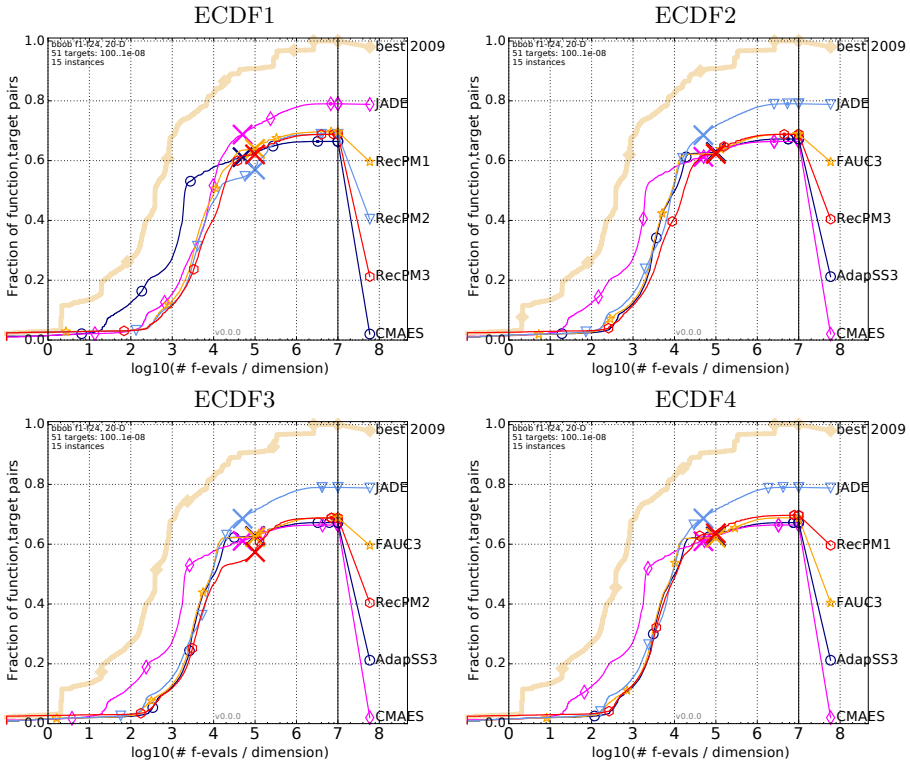

Fig. 1. Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions in 20-D. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers.

(NP) and crossover rate (CR), while discount factor (γ) and minimum selection probability ($p_{\min}$) belong to *RecPM-AOS*.

4 Experimental analysis

In the following, we compare the performance of proposed *RecPM-AOS* within DE with two other algorithms, namely *DE-F-AUC* [5] and *PM-AdapSS-DE* [7], for the online selection of mutation strategies in DE. More advanced DE variants are available in the literature, however, we want to understand and analyse the impact of the various AOS methods without possible interactions with other adaptive components of those variants. Nonetheless, for the sake of completeness, we also compare our results with two state-of-the-art algorithms *JADE* [17] and *CMAES* [10]. *JADE* is a DE variant that uses a mutation strategy called "current-to-pbest" and adapts the crossover probability CR and mutation factor F using the values which proved to be useful in recent generations. *CMAES* is

an evolution strategy that samples new candidate solutions from a multivariate Gaussian distribution and adapts its mean and covariance matrix.

We use BBOB (Black-box optimisation Benchmarking) [8] test suite to test the algorithms. BBOB provides an easy to use tool-chain for benchmarking black-box optimisation algorithms for continuous domains and to compare the performance of numerical black-box optimisation algorithms. We evaluate all algorithms on the 24 noiseless continuous benchmark functions [9] provided by BBOB, each with 15 different instances, totalling to 360 function instances. Each instance of a function is a rotation and/or translation of the original function leading to a different global optimum. These 24 functions are grouped in five classes, namely, separable, moderate, ill-conditioned, multi-modal and weak-structure functions. Each algorithm with AOS method is run to a maximum number of $10^5 \cdot d$ function evaluations (FEvals), where d is the dimension of the benchmark function. In this paper, we focus on $d = 20$ for all functions. The solutions in the initial population for each function instance are generated with different seeds.

4.1 Parameter tuning

We tune the parameters of the *DE-RecPM-AOS*, *DE-F-AUC* and *PM-AdapSS-DE* using the offline automatic configurator IRACE [12], which saves the hassle of manual tuning and allows for a fully specified and reproducible procedure. The input given to IRACE is the range of all parameters that need tuning (Table 2) and a set of training function instances; it then looks for good performing parameter configurations by executing the target algorithm on different training instances with a budget of 10^4 FEvals. In our case, the training set consist of only 10% of the function instances, randomly selected within each class, to avoid over-fitting.

In order to evaluate the impact of parameter tuning, we consider three parameter configurations of each algorithm. The first configuration is obtained by tuning all parameters of DE and the AOS methods. The second configuration is obtained by tuning only the parameters of the AOS methods, while the parameter values of DE are taken from [4]: $CR = 1.0$, $F = 0.5$ and $NP = 200$. The value $CR = 1.0$ means that a mutation strategy is applied to each dimension of all parents, which maximizes the impact of the mutation strategies. Finally, the third configuration (*default*) uses the settings suggested in [4] for *DE-F-AUC* and *PM-AdapSS-DE*, which uses the DE settings described earlier and AOS settings tuned with a different configurator. All parameter configurations are shown in Table 2.

4.2 Comparison of AOS methods with different parameter settings

After tuning, each obtained configuration is evaluated on the full BBOB benchmark set. We use plots of the Empirical Cumulative Distribution Function (ECDF) to assess their performance (Fig. 1). The ECDF displays the proportion of problems solved within a specified budget of function evaluations (FEvals) for different targets $f_{\text{target}} = f_{\text{opt}} + \Delta f$, where f_{opt} is an the optimum function value to reach with some precision $\Delta f \in [10^{-8}, 10^2]$. In the plots, FEvals is given on

Table 2. Optimal parameter configurations selected from the range shown below the parameter name. The following prefix abbreviations are used: RecPM for DE-RecPM-AOS, AdapSS for PM-AdapSS-DE and FAUC for DE-F-AUC. The symbol '-' in the table means that the parameter is not applicable to the AOS method.

	F $[0.1, 2.0]$	**NP** $[50, 400]$	**CR** $[0.1, 1.0]$	$\boldsymbol{\alpha}$ $[0.0, 1.0]$	$\boldsymbol{p}_{\min}$ $[0.0, 0.25]$	$\boldsymbol{\gamma}$ $[0.1, 1.0]$	**W** $[0, 200]$	**C** $[0.0, 1.0]$
RecPM1	0.47	168	0.98	-	0.17	0.75	-	-
RecPM2	0.5	200	1.0	-	0.11	0.46	-	-
RecPM3	0.5	200	1.0	-	0.0	0.6	-	-
AdapSS1	0.51	117	0.97	0.48	0.22		-	-
AdapSS2	0.5	200	1.0	0.86	0.04	-	-	-
AdapSS3	0.5	200	1.0	0.6	0.0	-	-	-
FAUC1	0.24	96	0.55	-	-	-	31	0.14
FAUC2	0.5	200	1.0	-	-	-	5	0.35
FAUC3	0.5	200	1.0	-	-	-	50	0.5

the x-axis and y-axis represents the fraction of problems solved. A large symbol '×' shows the maximum number of function evaluations given to each algorithm, in our case, $10^5 \cdot d$ FEvals are given to each algorithm with AOS method. Results reported after this symbol use bootstrapping to estimate the number of evaluations to reach a specific target for a problem [3]. The results denoted with `best 2009` correspond to the artificial best algorithm from the BBOB-2009 workshop constructed from the data of the algorithm with the smallest aRT (average Run Time) for each set of problems with the same function, dimension and target. The aRT is calculated as the ratio of the number of function evaluations for reaching the target value over successful runs (or trials), plus the maximum number of evaluations for unsuccessful runs, divided by the number of successful trials. Data to generate ECDF graphs for DE-F-AUC3, PM-AdapSS-DE3, CMAES and JADE is obtained directly from the COCO website.[3] The trials that reached f_{target} within specified budget are termed as successful trials, *#succ*. Table 3 shows the aRT (average Run Time), calculated as the ratio of the number of function evaluations for reaching the target value over successful runs, plus the maximum number of evaluations for unsuccessful trials, divided by the number of successful trials, on four out of 24 functions only due to limited space. The runtime for a function becomes undefined if there are no successful runs. The complete table can be seen in the supplementary material [15].

We expected to tune *DE-F-AUC* and *PM-AdapSS-DE* algorithms with the hope to replicate the original results for *DE-F-AUC3* and *PM-AdapSS-DE3* [5,7]. But we could not match the results shown in these papers. Thus, we decided to use the data available online at the COCO website and compare variants of proposed algorithm with *DE-F-AUC3* and *PM-AdapSS-DE3* only. The interested reader is referred to the supplementary material [15] to find the results of tuned *DE-F-AUC* and *PM-AdapSS-DE* algorithms. The ECDF graphs of variants of

[3] http://coco.gforge.inria.fr/doku.php?id=algorithms-bbob

proposed algorithm with *DE-F-AUC3* and *PM-AdapSS3* are shown in four plots of Fig. 1 that show the performance of algorithms averaged over all 360 functions tested. From now on we only talk about the original results and not the replicated ones.

ECDF1 shows results obtained for three variants of *DE-RecPM-AOS*. As expected, proposed algorithm with all tuned parameters outperformed its all other variants both in terms of speed and percentage of problems solved. When all three AOS methods use the default settings (ECDF2), it is estimated that *F-AUC* and *RecPM-AOS* solves the same number of problems but within given budget all algorithms solved same number of problems with varied speed. The third graph (ECDF3) where only parameters of AOS method are tuned in proposed algorithm shows that *DE-F-AUC3* and *PM-AdapSS-DE3* solves maximum problems with almost same speed within given budget. But when given more FEvals, according to bootstrapping technique, *DE-RecPM-AOS2* shows the same performance as *DE-F-AUC3* by solving the same number of problems whereas *PM-AdapSS-DE3* could not match the performance of other two algorithms. ECDF4 compares results of *DE-RecPM-AOS1*, *DE-F-AUC3* and *PM-AdapSS-DE3*. The proposed method with all tuned parameters that is, parameters of DE algorithm and of *RecPM-AOS* method outperformed all other algorithms by solving 75% of the problems. This is clearly because of the properties the proposed AOS method has. The tuned configurations of replicated algorithms: *DE-F-AUC* and *PM-AdapSS-DE* are not better than the original results reported, which we cannot replicate.

Summing up the above discussion, it can be said that tuning all the parameters of the proposed algorithm (*DE-RecPM-AOS1*) outperformed all its variants, thus tuning on training set plays an important role. It also outperformed all other AOS methods within DE solving 75% of the total problems. Thus, historical information preserving property in the form of reward and using Bellman equation to estimate quality of operator led to efficient adaptability of operators. On the other hand both F-AUC and RecPM-AOS makes use of past performances of operators, we do that by defining reward of each operator capturing a fraction of its last reward which reduces the hassle of maintaining a window of certain size. However, *F-AUC* and *PM-AdapSS* shows similar speed in solving a fixed number of problems and *DE-RecPM-AOS1* has faster convergence speed and increased percentage of problems solved. The full table showing aRT for AOS methods within DE algorithm in supplementary material shows that no one algorithm has best converging speed for all functions and *DE-F-AUC3* and *DE-RecPM-AOS1* shows competitive results.

4.3 Comparison of *RecPM-AOS* with state-of-the-art algorithms

CMAES and JADE are given a budget of $5 \cdot 10^4$ FEvals. When comparing different versions of *DE-RecPM-AOS* with *CMAES* and *JADE*, proposed algorithm with all tuned parameters is able to solve most functions than *CMAES* as seen in ECDF4 in figure 1 that is, almost 10% more than best version of *DE-RecPM-AOS*: *DE-RecPM-AOS1*. However, *JADE* manages to solve majority of

Table 3. Average runtime (aRT in number of function evaluations) divided by the respective best aRT measured during BBOB-2009 in dimension 20. RecPM: DE-RecPM-AOS, AdapSS: PM-AdapSS-DE, FAUC: DE-F-AUC. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{\text{opt}}+10^{-8}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Best results are printed in bold.

20-D

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	43	43	43	43	43	43	43	15/15
AdapSS3	101	196	291	377	465	646	827	15/15
CMAES	**7.5**	**13**	**20**	**26**	**33**	**45**	**58**	15/15
FAUC3	93	180	265	350	431	597	763	15/15
JADE	47	94	143	191	240	340	437	15/15
RecPM1	96	187	278	369	459	636	819	15/15
RecPM2	114	219	317	410	510	707	932	15/15
RecPM3	132	263	432	932	1234	1621	2076	14/15
f2	385	386	387	388	390	391	393	15/15
AdapSS3	52	63	73	83	93	112	132	15/15
CMAES	36	43	45	47	**47**	**48**	**50**	15/15
FAUC3	48	58	68	**77**	86	105	123	15/15
JADE	**28**	**34**	**39**	**44**	50	61	71	15/15
RecPM1	47	57	66	76	85	103	121	15/15
RecPM2	72	90	108	127	147	186	223	15/15
RecPM3	66	111	180	205	278	333	360	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f13	652	2021	2751	3507	18749	24455	30201	15/15
AdapSS3	28	13	12	12	2.6	2.6	2.6	15/15
CMAES	**6.3**	**5.1**	**4.5**	**4.4**	**1.9**	4.6	8.4	12/15
FAUC3	25	12	11	11	2.4	**2.5**	**2.5**	15/15
JADE	17	14	15	14	3.6	4.8	9.0	15/15
RecPM1	29	14	14	14	3.2	3.6	4.2	15/15
RecPM2	36	17	16	16	3.6	4.1	7.2	15/15
RecPM3	44	32	32	38	7.8	8.8	8.0	15/15
f14	75	239	304	451	932	1648	15661	15/15
AdapSS3	43	34	43	40	25	20	2.8	15/15
CMAES	**4.2**	**3.0**	**3.7**	**4.3**	**4.2**	**6.2**	**1.2**	15/15
FAUC3	33	30	38	36	23	19	2.8	15/15
JADE	18	18	23	25	20	38	62	5/15
RecPM1	40	33	42	41	26	24	4.2	15/15
RecPM2	52	44	58	53	32	27	4.3	15/15
RecPM3	47	43	54	51	35	38	5.1	15/15

the problems than any AOS methods within DE, shown in ECDF1. In the initial runs, *CMAES* has faster convergence speed than any other algorithm.

5 Conclusion and future work

We proposed a variant of probability matching, *recursive-PM*, as a parameter control method that gives the quality as an aggregated estimate of future performances of operators. The proposed adaptive operator selector adaptively selects a mutation strategy in Differential Evolution. The algorithm differs from classical PM in the way it assigns the quality to a strategy. The reward given to an operator depends on the short term success of that operator. It is compared with two AOS methods *DE-F-AUC*, *PM-AdapSS-DE* and two state-of-the-art algorithms *CMAES*, *JADE*. The overall performance of *Recursive-PM* within DE with tuned parameters shows that it outperforms other two AOS methods that is, *DE-F-AUC* and *PM-AdapSS-DE*, and *CMAES* by solving 75% of the problems. The proposed algorithm could not outperform *JADE*, but had similar convergence rate.

IRACE is used to find the good offline settings for the proposed AOS method, which illustrates the usefulness of offline procedures to successfully design new online adaptation methods. It is used to train the parameters on 10% of the total function instances. We plan to extend the proposed algorithm by integrating it with different definitions of credit assignment to compete with the state of the art algorithms. To make *RecPM-AOS* perform better, we plan to extend proposed algorithm by finding and tuning more parameters involved in the method such as the fraction of previous reward to take under consideration when designing credit assignment technique.

References

1. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning **47**(2-3), 235–256 (2002)
2. Das, S., Mullick, S.S., Suganthan, P.N.: Recent advances in differential evolution–An updated survey. Swarm and Evolutionary Computation **27**, 1–30 (2016)
3. Efron, B., Tibshirani, R.J.: An introduction to the bootstrap. CRC press (1994)
4. Fialho, Á., Schoenauer, M., Sebag, M.: Fitness-AUC bandit adaptive strategy selection vs. the probability matching one within differential evolution: an empirical comparison on the BBOB-2010 noiseless testbed. In: Pelikan, M., et al. (eds.) GECCO (Companion), pp. 1535–1542 (2010)
5. Fialho, Á., Schoenauer, M., Sebag, M.: Toward comparison-based adaptive operator selection. In: Pelikan, et al. (eds.) Proc. of the Genetic and Evolutionary Computation Conf. (GECCO) 2010, pp. 767–774. Portland, Oregon, USA (2010)
6. Goldberg, D.E.: Probability matching, the magnitude of reinforcement, and classifier system bidding. Machine Learning **5**(4), 407–425 (1990)
7. Gong, W., Fialho, Á., Cai, Z.: Adaptive strategy selection in differential evolution. In: Pelikan, M., et al. (eds.) GECCO, pp. 409–416 (2010)
8. Hansen, N., Auger, A., Mersmann, O., Tusar, T., Brockhoff, D.: COCO: A platform for comparing continuous optimizers in a black-box setting. Arxiv preprint arXiv:1603.08785 (2016)
9. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Tech. Rep. RR-6829, INRIA, France (2009), updated February 2010
10. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation **9**(2), 159–195 (2001)
11. Karafotias, G., Hoogendoorn, M., Eiben, A.E.: Parameter control in evolutionary algorithms: Trends and challenges. IEEE Transactions on Evolutionary Computation **19**(2), 167–187 (2015)
12. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., Birattari, M.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016)
13. Price, K., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Springer, New York, NY (2005)
14. Rust, J.: Structural estimation of Markov decision processes. In: Handbook of Econometrics, vol. 4, pp. 3081–3143. Elsevier (1994)
15. Sharma, M., López-Ibáñez, M., Kazakov, D.: Performance assessment of recursive probability matching for adaptive operator selection in differential evolution: Supplementary material. https://github.com/mudita11/AOS-comparisons (2018). https://doi.org/10.5281/zenodo.1257672
16. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)
17. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation **13**(5), 945–958 (2009)