# Toward Comparison-based Adaptive Operator Selection

Submitted to the *Genetic Algorithms* track

## ABSTRACT

Adaptive Operator Selection (AOS) turns the impacts of the applications of variation operators into Operator Selection through a Credit Assignment mechanism. However, most Credit Assignment make direct use of the fitness gain between parent and offspring. A first issue is that the Operator Selection that uses such Credit Assignment is likely to be highly dependent on the a priori unknown bounds of the fitness function. Additionally, those bounds are likely to change along evolution, as fitness gains tend to get smaller as convergence occurs. Furthermore, and maybe more importantly, a fitness-based credit assignment forbid any invariance by monotonous transformation of the fitness that is a source of robustness for comparison-based Evolutionary Algorithms. In this context, this paper proposes two new Credit Assignment mechanisms, one inspired by the Area Under the Curve paradigm, and the other close to the Sum of Ranks. Using fitness improvement as raw reward, and directly coupled to a Multi-Armed Bandit Operator Selection Rule, the resulting AOS obtain very good performances on both the OneMax problem and some artificial scenarios, while demonstrating their robustness with respect to hyper-parameter and fitness transformations. Furthermore, using fitness ranks as raw reward results in a fully comparison-based AOS with reasonable performances.

## Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence: Problem Solving, Control Methods, and Search

## General Terms

Algorithms

## Keywords

Parameter Control, Adaptive Operator Selection, ROC Area Under Curve, Multi-Armed Bandits

## 1. INTRODUCTION

Invariance is an ubiquitous concept of science. From Lavoisier's first statement of the mass/energy equivalence to Einstein relativity principles, conservation laws have been one basis of many scientific theories, and invariances generally lead to conservation laws. Mathematically speaking, invariances with respect to some transformations allow the mathematician to generalize properties from one object to the whole class of objects that is generated by the transformations. For instance, distances are invariant under orthogonal transformations, and this lead to Euclidian geometry.

In the realm of Evolutionary Computation, the importance of invariance has been stressed, too. Some Evolutionary Algorithms, for instance, are invariant under monotonous transformations of the fitness function, by the of comparison-based selection mechanisms (e.g., deterministic, rank-based, or tournament selection). Whereas this is known in practice to add robustness to the algorithm (e.g. protection against super-individuals that would quickly invade the population in case of proportional selection), it also leads to theoretical results that immediately apply to any comparison-based algorithm (evolutionary or not) [12]. The best possible consequence of invariances within the EC community is illustrated by the success of CMA-ES algorithm [14]: CMA-ES is not only comparison-based, and hence invariant by monotonous transformations of the fitness function, but also invariant under orthogonal transformations of the search space, and as a consequence performs similarly on a given function and on all its rotated versions, outperforming most of its challengers on non-separable functions during the BBOB workshop at GECCO 2009[1]. Such invariance also allowed its author to come up with a very robust setting for the CMA-ES internal parameters [14]. But apart from this notable exception in the continuous case, parameter tuning is known to be one of the main drawbacks of EAs, as no theoretical guideline exists to help the practitioner, and because all great successes of EAs have demonstrated that different problems require different parameters even for the same algorithm.

This paper is concerned with the possible invariance properties of *Adaptive Operator Selection* (AOS) in the framework of EAs. AOS deals with the on-line choice among available variation operators during an Evolutionary run, and involves a selection mechanism that uses statistics on raw rewards that have been gathered after previous applications of these operators. Most AOS methods proposed up to now use as reward, or as part of the reward, the fitness im-

[1]`http://coco.gforge.inria.fr/doku.php?id=bbob-2009-results`

provement, i.e. the progress in fitness of the offspring compared to its parents. This raises several issues: on the one hand, fitness-dependent reward needs to be scaled anew for each problem whose fitness range is not known, and this affects the tuning of the meta-parameters of the AOS method. Secondly, in any case, the fitness gains generally decrease as evolution proceeds, and no static tuning can handle properly the complete range of fitness gains along an evolutionary run. And thirdly, the robustness of the algorithm resulting from its eventual invariance with respect to monotonous transformations of the fitness cannot hold any more.

In order to address those issues, new *Credit Assignment* mechanisms are proposed, that rely as less as possible on actual fitness values. One way to achieve this, as done for the selection mechanisms, is to ground the credit assignment on the relative ranking of the raw rewards with respect to one-another. Furthermore, using fitness ranks as raw rewards results in a true comparison-based AOS mechanism.

The paper is organized in the following way. Section 2 presents a brief overview of the state-of-the-art of AOS techniques, and discusses them from the point of view of invariance properties and meta-parameter setting. Building on those observations, Section 3 proposes two *Credit Assignment* mechanisms that only use ranks of impact measures. When coupled with an impact measure directly based on the rank of the newborn offspring in the population, the resulting AOS schemes achieve complete invariance with respect to monotonous transformations of the fitness. Section 4 details the experimental validation of the proposed methods on the OneMax toy problem with respect to transformations of the fitness function, and on artificial scenarios for comparison with other existing AOS. Section 5 discusses some further research that is opened by this work, and the paper ends with the conclusions in Section 6.

## 2. ADAPTIVE OPERATOR SELECTION

This paper is concerned with the on-line adaptation of the parameters of an EA, aka *adaptive parameter control* in the literature [7, 8], and will not discuss any off-line setting, aka *parameter tuning*. One strong argument for on-line parameter control is that, as the algorithm proceeds from a global (early) exploration of the landscape to a more focused, exploitation-like behavior, the parameters should be continuously adjusted according to the current needs of the search. It has been empirically and theoretically demonstrated that different values of parameters might be optimal at different stages of the search process (see [8, p.21] and references therein). For example, usually a fixed application rate is defined by the user for each variation operator, what is always sub-optimal, as in fact, following the same *rationale*, exploration-like operators (e.g. n-points crossover) should be favored in the initial stages of the search, while more exploitation-like operators should be used in the later stages.

Amongst parameter control methods, 3 categories are usually distinguished [8]: *deterministic* control follows some pre-defined deterministic rules, and thus de facto amounts to off-line setting of such rule; *self-adaptive* techniques rely on random modifications of the parameters, by letting evolution itself control their values “for free”, however requiring the exploration of both the search space of the variables and that of the parameter values; *adaptive* methods modify the parameter values based on the informations given by the search itself, *i.e.*, it monitors some particular properties of the evolutionary process and use changes in these properties as an input signal to change the parameter values.

Included in this latter category, *Adaptive Operator Selection* (AOS) aims at autonomously selecting which of the available variation operators should be applied at a given time, based on the history of the current search. Following [11], we shall distinguish two phases, the *Credit Assignment* mechanism, that is used to turn the observed impact of the application of a given operator into a *reward*, and the *Operator Selection Rule*, that actually selects one operator from the rewards gathered by each operators in the past.

### 2.1 Credit Assignment

Starting from Davis' seminal work [6], several approaches have been proposed for *Credit Assignment*, differing in the way the impact of an operator is measured, which operator the reward will be awarded to, and how the rewards are accumulated along time for each operator, to be used by the next *Operator Selection Rule*.

Most methods use as raw reward some fitness improvement of the newborn offspring compared to a base individual, that might be (i) its parent [18, 22, 2, 9, 11], (ii) the current best in the population [6], (iii) or the median individual of the current population [17]. If no improvement is achieved, usually a null reward is assigned to the operator. A recent proposal, in the case of highly multimodal problems, aggregated the fitness gain and the variation of some measure of diversity, in a mechanism termed *Compass* [19]. Because this paper is concerned with robustness against fitness transformations, that has no relationship with any measure of diversity, such approach will not be considered here. However, all results of the present paper could be applied in turn to the fitness-based part of the *Compass Credit Assignment*.

With respect to which operators to assign the credit to, most previous works cited above only consider the operator that generated the newborn offspring. Some authors, however, propose to assign credit to the operators used to generate the ancestors of the current individual (*e.g.*, using some bucket brigade-like algorithm [6, 17]), based on the claim that the existence of efficient parents is indeed as important as the creation of improved offspring. Others, however, more recent work suggested that rewarding the ancestors' operators sometimes degrades the results [2].

Finally, the *Credit Assignment* transforms, for each operator, the raw reward into the actual reward that will be used by the *Operator Selection Rule* to choose one operator for next offspring generation. The existing approaches here differ in the statistics that are considered in order to compute such reward. Most methods only consider the most recent operator application. Others use the average of the raw rewards achieved over a few applications of the operators. More recently, initialized in [23], the use of the extreme value (statistical outlier) over a few applications was proposed, based on the idea that highly beneficial but rare events might be better to the search than regular but smaller improvements. The reported comparative results with other *Credit Assignment* mechanisms show the superiority of this approach, over a set of continuous benchmark problems [23], and in the GA context [9, 11].

### 2.2 Operator Selection Rules

Most *Operator Selection Rule* rules attach an empirical quality to each operator, and two families of such *Operator Selection Rule* are considered in this paper. The probability-based methods *Probability Matching* (PM) [13, 18, 2] and *Adaptive Pursuit* (AP) [20, 21] use the empirical qualities to update operator probabilities[2], and then use those probabilities to select an operator by roulette wheel. The bandit-based methods (*Multi-Armed Bandit* (MAB), *Dynamic Multi-Armed Bandit* (DMAB) [5, 9, 10, 11] use variants of the Multi-Armed bandit paradigms [1] to deterministically choose one operator from the empirical quality (termed empirical reward in the bandit terminology).

### *2.2.1 Probability-based Operator Selection*

Both PM and AP methods compute the empirical quality of operator $i$, noted $\hat{p}_{i,t}$, using some relaxation factor $\alpha \in ]0,1])$: if operator $i$ is applied and receives reward $r_t$, its quality is updated by the following formula

$$\hat{p}_{i,t+1} = (1-\alpha)\hat{p}_{i,t} + \alpha\, r_t \tag{1}$$

From thereon, the most common and straightforward way of doing *Operator Selection Rule* is the PM method [13, 18, 2] that computes from the empirical qualities the probabilities of applying each operator, with the constraint that no probability can be lower than a user-defined minimal value $p_{min}$. In particular, because all probabilities sum up to one, the maximum value of the probability of any operator is $p_{max} = 1 - K * p_{min})$ if there are $K$ available operators. This badly impacts the performances of the method [20, 9]. The formal update of operator selection probability $s_{i,t}$ from empirical quality reads

$$s_{i,t+1} = p_{min} + (1 - K * p_{min})\frac{\hat{p}_{i,t+1}}{\sum_{j=1}^{K}\hat{p}_{j,t+1}} \tag{2}$$

Originally proposed for learning automata, the AP (AP) [20, 21] is an *Operator Selection Rule* technique that partly addresses this drawback by implementing a winner-takes-all strategy from the quality measure: the operator with maximum empirical quality is applied, and its probability is moved toward $p_{max}$ while those of all other operators are decreased equally so that the sum remains 1:

$$\begin{cases} i^* & = argmax\{\hat{p}_{i,t}, i = 1 \dots K\} \\ s_{i^*,t+1} & = s_{i^*,t} + \beta\,(p_{max} - s_{i^*,t})\,, \\ s_{i,t+1} & = s_{i,t} + \beta\,(p_{min} - s_{i,t})\,, \text{ for } i \neq i^* \end{cases} \tag{3}$$

Though it also relies on a user-defined lower bound $p_{min}$ to maintain a minimal level of exploration, another user-defined parameter, $\beta$, is used to control the greediness of the strategy, *i.e.*, how fast the probability of selecting the current best will converge to $p_{max}$ while all the others will go to $p_{min}$. AP thus obtains better results than PM in most, if not all, reported works [21, 10, 11].

### *2.2.2 Bandit-based Operator Selection*

Bandit algorithms have been proposed to solve the *Exploration vs. Exploitation* (EvE) dilemma in a general context. When facing $k$ independent arms, with unknown boolean reward distribution, the *Upper Confidence Bound* (UCB) strategy, proposed by Auer et al. [1], provides guarantees of asymptotic optimality in terms of cumulated reward, being phrased as *"Optimism in front of the Unknown"*. The empirical quality is the actual average of past rewards, and confidence bounds are computed, based on passed frequency of use of the arm. Next arm to pull is the one with the highest possible value within those bounds. Within AOS, each operator is considered as an arm, and the reward is the one that is provided by the *Credit Assignment*. At given time $t$, denote $n_{i,t}$ the number of times the $i$-th arm has been tried. The update of the empirical reward can be written as [5]

$$\hat{p}_{i,t+1} = (1 - \frac{1}{n_{i,t}})\hat{p}_{i,t} + \frac{1}{n_{i,t}}\, r_t \tag{4}$$

and the adapted UCB formula reads

$$\hat{p}_{j,t} + C\sqrt{\frac{2\log\sum_k n_{k,t}}{n_{j,t}}} \tag{5}$$

where C is a user-defined constant needed to account for the range of the rewards: in the original UCB, rewards are either 0 or 1, and empirical reward lies in $[0,1]$. However, the scale of the rewards is unknown in the AOS framework, and careful tuning of $C$ is required [10, 11].

Furthermore, the original UCB framework is static, i.e., it is assumed that the reward distribution of all arms remains the same all along the experiment. This is obviously not the case, and the original MAB algorithm, though it theoretically keeps exploring all arms, will asymptotically take far too much time to detect something has changed in the distributions. This motivated the design of the *Dynamic Multi-Armed Bandit* algorithm [15], that adds to the original MAB the Page-Hinkley change detection statistical test [16]: when this test triggers, the Multi-Armed Bandit is restarted from scratch.

Formally, the PH test works as follows, where $\bar{r}_t$ denotes the average reward over the last $t$ steps, and $e_t$ the difference between the instant and average reward, plus some small tolerance $\delta$. Considering the random variable $m_t = \sum_1^t e_i$, the PH test is triggered when the difference between $M_t = \max_{i \leq t} |m_i|$ and $|m_t|$ is greater than some user-specified threshold $\gamma$:

$$\bar{r}_t = \tfrac{1}{t}\sum_{i=1}^{t} r_i \qquad e_t = (r_t - \bar{r}_t + 0.15) \qquad m_t = \sum_{i=1}^{t} e_i$$

$$\text{Return } (\max_{i=1\dots t}\{|m_i|\} - |m_t| \, > \gamma) \tag{6}$$

Parameter $\gamma$ controls the sensitivity of the test, and thus the rate of false alarms.

## 2.3 Robustness and Invariance Properties

When considering the AOS mechanisms described above (PM, AP, MAB, and DMAB) with respect to their robustness under transformations of the fitness functions, two aspects have to be distinguished.

First, all those techniques use as reward some fitness improvement of the newborn offspring with respect to its parent (but it could also be with respect to any individual, or average in the population). This already implies an invariance with respect to translations of the fitness (i.e., $\mathcal{F} \rightarrow \mathcal{F} + a$ for some real value $a$). This is obvious, but for instance not even the case with the standard roulette-wheel selection of classical GAs.

Looking further, a more important difference can be seen between PM and AP on the one side, and the bandit-based

[2]Methods that recompute those probabilities from scratch from the most recent rewards [17, 22] will not be considered here.

algorithms on the other side. Both PM and AP are invariant by linear scaling of the fitness function (i.e., $\mathcal{F} \rightarrow a \times \mathcal{F}$ for some real value $a > 0$). They are, however, not invariant under general monotonous transformation of the fitness, as the actual value of the fitness gains may greatly vary with such a transformation, possibly resulting in very different behavior of the AOS. Ideally, this should be compensated by the learning parameter $\alpha$ – though in practice the same settings are generally used. Experimental results presented in Section 4.3 will shed more light on this.

On the other hand, for all MAB-based AOS mechanisms [5, 9], the actual value of the fitness gains are directly used in the UCB formula (5). It can be of course argued that any linear transformation of the fitness can be easily compensated by an equivalent transformation of the scaling constant $C$. Indeed – but then $C$ plays two radically different roles here: accounting for the scale of the fitness, and tuning the balance between exploitation and exploration. This makes it a very sensitive parameter. Furthermore, for the DMAB AOS, the additional parameter $\gamma$ is also very sensitive, and very dependent on actual reward values.

In order to try to alleviate this issue, a normalization done "on-the-fly" according to the highest reward recently received was later proposed in [11], but did not seem to improve the results. Indeed, as the normalization factor depends on the region of the landscape that is currently being explored, the same gain might have different weights in the update of the empirical estimates throughout the search process. These issues lead to extremely problem-dependent meta-parameter for the bandit-based AOS.

The rest of the paper is devoted to propose alternative AOS mechanisms to avoid the drawbacks discussed above.

## 3. IMPROVING ROBUSTNESS OF MAB AOS

This Section will present two variants of the MAB-based AOS that address the issues about robustness with respect to fitness transformations listed above. First, two original *Credit Assignment* are proposed, using only the ranks of the raw rewards measuring the recent impacts of the operators, i.e., the fitness improvements for all AOS considered here. In order to go further toward complete invariance under monotonous transformation of the fitness, a rank-based measure of impact (raw reward) is then argued.

### 3.1 Credit Assignment from Reward Ranks

The first modification proposed here aims at removing the dependency of the MAB AOS with respect to the actual scaling of the fitness function, or, equivalently, the double role of parameter $C$ in UCB formula (5) discussed in Section 2.3. To this aim, the idea is to replace the statistics made on the raw rewards values (i.e., fitness improvement), by statistics made on their ranks.

A sliding window of size $W$ is used to store the impacts of the previous $W$ applications of operators. However, a major difference with the windows used in previous studies [9, 11] is that here the results of all operator applications are stored in the same window, whereas in [9, 11], there was one window per operator. EAch slot in the windows thus contain here one fitness improvement and the operator that was responsible for this improvement.

Two approaches are then proposed to compute some reward from the ranked impact measures.

The **Sum of Ranks** method assigns some *rank-value* to each position in the window. Initial setting assigned rank-value $W-1$ to the highest position in the sorted list, $W-2$ to next one, ..., and more generally, rank-value $W-r$ to the $r^{th}$ position. The credit for each operator is then computed by summing up the rank-values of all positions that correspond to its own application. However, it soon turned out that one additional parameter was needed, to allow for some memory decay, and possibly increase the influence of the top-ranked operators. Let $D \in ]0,1]$ be the decay factor. Then position $r$ in the window is assigned rank-value $D^r(W-r)$.

The **AUC** method borrows the idea of the Area Under the Curve, that is used as a criterion in some Machine Leaning applications to compare binary classifications rules [4]. Computing the reward of a given operator amounts to go down the sorted list, and drawing, starting from the origin, the *Receiving Operator Curve* by adding a vertical segment each time the operator under scrutiny is found in the list, and an horizontal segment for all other operators. The Area Under the Curve is then normalized in such a way that it would be 1 if the target operator had occupied all top position in the list.

Figure 1 illustrates this (without decay in this case, for the sake of clearness) for a sample situation involving 2 operators, + and - respectively, and where the sorted list would be something like (+ - + + - - [- - +] + - - - +), with [- - +] meaning that those 3 positions have the same raw reward, leading to the diagonal line between points (3 3) and (5 4) (dotted lines are spaced by 1). The AUC reward is the grey area, normalized such that the total rectangle, here of area $7 \times 6$ has area 1. The ROC is the solid line, upper bound of this area. In case of decay, each segment of the ROC would have a length of size $D^r(W-r)$, the dotted lines would be moved accordingly, and the normalization would of course change in consequence, in order to keep the total AUC 1.

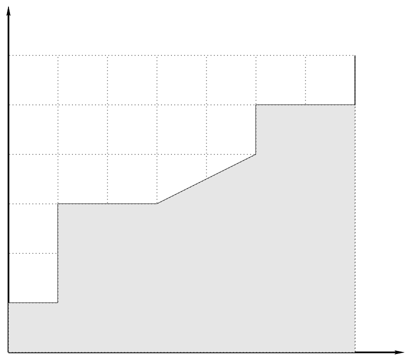

**Figure 1: Sample computation of AUC reward.**

However, it turned out that using either quantities (Sum-of-Ranks or Area Under the Curve) as a reward to be fed in one of the *Operator Selection Rule* described in Section 2.2 was not efficient at all, probably because those quantities are already statistics over some time window that are computed for all operators as soon as one is applied. On the other hand, using those quantities as empirical rewards directly in the UCB formula (5) proved successful. The resulting AOS mechanisms are called respectively *Sum of Rank Bandit* (SR-B) and *AUC-Bandit* (AUC-B).

### 3.2 Comparison-Based Raw Reward

Both SR-B and AUC-B AOSs are invariant with respect to linear fitness scaling, and behave similarly to PM and AP when facing some fitness transformation. Nevertheless, because the raw rewards that are used here are actual values of fitness improvements, some monotonous transformation will indeed modify the ranking of such values, and hence the outcome of the whole algorithm.

However, regarded as *Credit Assignment*s, both have been designed in order to compute a reward based on a ranked list – namely, the ranks of the fitness improvements brought by the operators. Replacing the fitness improvements by the raw fitnesses of the newborn offspring, those *Credit Assignment* allow us to compute a reward that is fully comparison-based, as only sorting some fitness values is required.

These two AOS will be termed F-SR-B and F-AUC-B respectively. Note that they are identical to SR-B and AUC-B respectively when run on artificial scenarios, because these scenarios assume a known reward. Hence experiments with artificial scenarios will only involve SR-B and AUC-B (Section 4.4). However, when used on an actual optimization problem (e.g., the OneMax problem in Section 4.3), both SR-B (resp. AUC-B) and F-SR-B (resp. F-AUC-B) should be distinguished.

## 4. EXPERIMENTAL RESULTS

This Section details some experimental validation of the proposed AUC-based and Sum-of-Ranks-based approaches. Because it was one motivation of the present work, their behavior with respect to transformation of the fitness will be compared to that of previously proposed AOS, and on the well-controlled experimental benchmark offered by the OneMax problem [9]. Then, more extensive comparisons will be performed on artificial scenarios, as proposed in [21, 5].

### 4.1 Scenarios

The well-known **OneMax problem**, aka the "drosophila of EC", has already been used to assess the performance of probability-based and bandit-based AOS [9, 10]. As in these works, the 10000-bits problem is considered, and the available variation operators are mutation operators, ranging from the standard bit-flip operator (every bit is flipped with probability $1/N$) to the $b$-bit mutations (flipping exactly $b$ randomly chosen bits) with $b = 1, 3, 5$. Two scenarios are used, the first one using only the $1-$ and $5 - bits$ mutations, and the second one with the 4 listed mutations. The EA is a $(1 + 50) - EA$, i.e., one parent gives birth to 50 offspring by mutation, and the best of the 51 individuals becomes the parent of next generation. The performance is given by the number of generations needed to reach the optimum: even with 10000 bits, this is a very easy problem and any decent parameter setting will reach the optimum value. However, despite being far from any real-world situation, this problem offers a completely known and mastered experimental testbed, e.g., the optimal strategy for operator choice is completely known [9].

**Artificial Scenarios** are another popular setting to compare AOS, initiated with Thierens' original benchmark [20]. A set of 5 "operators" have prescribed reward distribution, that varies along successive epochs ($\Delta T$ time steps). During every epoch, the operator reward is uniformly drawn in some interval: $[4, 6]$ for the current best operator, $[3, 5]$ for the second best, and so forth, until $[0, 2]$ for the worst operator (since these intervals overlap, the second best operator occasionally gets better rewards than the best one, etc). At the end of every epoch, the reward distributions are permuted, using pre-defined permutations[3]. The performance associated to an *Adaptive Operator Selection* is the cumulative reward obtained during this sequence of 10 epochs.

Within this benchmark, thereafter referred to as *Uniform*, an operator always gets some positive reward. Still, in an actual evolutionary context, an operator would most often bring no improvement at all (after the first generations), thus providing the *Adaptive Operator Selection* with no information whatsoever. For this reason, two variants of the *Uniform* benchmark respectively referred to as *Boolean* and *Outlier*, have been proposed in [5].

In the *Boolean* scenario, the best operator gets a reward of 10 with probability 50% (and 0 otherwise); it thus has same reward expectation as in the Uniform scenario, though with a much higher variance. The second best operator gets a reward of 10 with probability 40% and 0 otherwise, and so forth, until the worst operator, getting a reward of 10 with probability 10% and 0 otherwise. In this scenario, operators only differ by their probability of getting a non-null reward; the reward takes the same value in all cases.

Quite the contrary, in the *Outlier* scenario all operators get a non-null reward with same probability (10%); the difference lies in the reward value, set to 50 for the best operator, 40 for the second best and so forth. While the reward expectation is still the same as in the Uniform benchmark, the *Adaptive Operator Selection* is provided with much less information (only 10% of the trials produce some information) and the reward variance is much higher than in the previous Boolean scenario.

The *Adaptive Operator Selection* ability to match the dynamics of evolution is assessed by varying the length of the epoch, set to $\Delta T = 200$ (respectively $\Delta T = 2000$) for fast (resp. slow) dynamics. As the reward expectation of the best operator is 5 in all scenarios, the maximal cumulative reward is 10,000 in the fast case and 100,000 in the slow one ($5 \times 10 \times \Delta T$).

### 4.2 Hyper-Parameter Setting

| Heuristic | H-P | Range | Comments |
|---|---|---|---|
| All previous | CA | X{Abs,Nor},Av{Abs,Nor} | *Credit Assignment* |
| All | $\mathcal{W}$ | $\{10, 50, 100, 500\}$ | Window size |
| AUC, SR | $\mathcal{D}$ | $\{.1, .3, .5, .7, .9, 1\}$ | Decay factor |
| AP, PM | $p_{min}$ | $\{0; .05; .1; .2\}$ | Min. prob. |
| AP, PM | $\alpha$ | $\{.1, .3, .6, .9\}$ | Adaptation rate |
| AP | $\beta$ | $\{.1, .3, .6, .9\}$ | Learning rate |
| DMAB(PH) | $\gamma$ | $Range(\mathcal{C}), \{250, 500, 1000\}$ | PH threshold |
| All bandit | $\mathcal{C}$ | $\{\{1, 5\}.10^{\{-4 \leq i \leq 2\}}\}$ | Scaling factor |

**Table 1: AOS Hyper-parameters and value range**

As previously mentioned, every *Credit Assignment* and *Operator Selection Rule* has some hyper-parameters that need to be set by the user. They are recalled in Table 1, together with the different discrete values that were tested. Indeed, in order to promote a fair comparison between all

[3]These permutations are: 41203 → 01234 → 24301 → 12043 → 41230 → 31420 → 04213 → 23104 → 14302 → 40213. More precisely, the best operator in the first epoch is the $op_4$, which becomes the worst one in the second epoch. The best operator in the second epoch is $op_0$, which was the fourth one in the first epoch.

competing AOS, all the results presented here were obtained using the best configuration for each technique among the ones that can be obtained from these ranges. However, rather than a full factorial Design Of Experiment, a Racing procedure was used, as advocated in [11]. The basic idea of Racing, introduced in the field of Evolutionary Computation in [3], is to start running some standard Design Of Experiment, but to stop wasting time testing parameter configurations that are statistically proved to be, at their best, worse than the best configuration to-date. Friedman race (F-Race) is used here, with confidence level 95% as advocated in [3], the elimination of unpromising runs starts after 11 runs, and the racing is stopped as soon as a single configuration remains, or after 50 runs of the remaining configurations have been run. The criterion used for its statistical test, applied with a confidence level of 95%, is the minimization of the number of generations needed to reach the optimal solution in the case of the OneMax and related functions, and the maximization of the cumulative reward for the artificial scenarios.

Let us make a few remarks about Table 1. First, the first parameter in this has not been described yet: the Credit Assignment for all methods from earlier work (PM, AP, MAB, and DMAB). As in [11], the actual reward that the AOS computes from the raw reward (the fitness improvement in the OneMax scenario, the pre-defined reward in the artificial scenarios) can be either the Average or the Extreme (*Avg* or *X* in Table 1) of all Absolute or Normalized values (*Abs* or *Nor* in the table) taken over the sliding time window of size $W$. In fact, independent Racing have been performed for the 4 *Credit Assignment* variants. Also note that 14 values are tested for the $C$ parameter for all bandit-based techniques (i.e., all except PM and AP), even though it does not have exactly the same meaning for MAB and DMAB than for the new approaches proposed here, as discussed in Section 2.3. In summary, and due to the 4 values of *Credit Assignment*, the number of different candidate parameter configurations that were tried in the Racing for the different techniques ranges from $4 \times 64$ for PM and $4 \times 256$ for AP to $4 \times 56$ for MAB and $4 \times 952$ for DMAB, due to the high uncertainty (and high sensitivity) of the Page Hinkley parameter $\gamma$. For the newly proposed techniques, 'only' 336 configurations needed to be tested (even when trying out all 14 values for $C$).

## 4.3 Robustness Results

A first series of experiments was conducted regarding the robustness of the different AOS with respect to some nonlinear transformations of the fitness, as discussed in Section 2.3. Every AOS discussed in this paper was run on the OneMax problem, and the Racing procedure described in above Section 4.2 was used to find out the best parameter configuration. Then this best parameter configuration was used on different transformations of the OneMax fitness $\mathcal{F}$, namely $\log(\mathcal{F})$, $\exp(\mathcal{F})$, and $\mathcal{F}^2$.

Complete results can be found on Table 4.3. Those results confirm the a priori discussion of Section 2.3: PM and AP are indeed much less sensitive to such nonlinear transformations of the fitness. Nevertheless, their performance is slightly degraded by these transformations. Because OneMax is a very easy problem, it remains of good quality, but running a complete Racing for XAbs AP using $\exp(\mathcal{F})$ results in a completely different optimal setting. In fact, the optimal setting for $\mathcal{F}$ ($W = 500$, $P_{min} = 0$, $\alpha = 0.9$, $\beta = 0.1$) is killed in the first elimination round during the Racing with $\exp(\mathcal{F})$, and the winner of the race $\exp(\mathcal{F})$ sets $W = 100$ and $\beta = 0.9$: because the $\exp(\mathcal{F})$ function varies much faster than the original $\mathcal{F}$, the probabilities need to be adjusted much more rapidly. Such result clearly demonstrates that the non-invariance under nonlinear transformation could eventually cause some serious loss of efficiency for more difficult problems.

**Table 2: Analysis of invariance w.r.t the OneMax function ($\mathcal{F} = \sum n$) and 3 monotonous transformations of it: $\log(\mathcal{F})$, $\exp(\mathcal{F})$ and $(\mathcal{F})^2$. The results on the transformed functions are omitted;** $Avg(all)$ **shows the average performance over 50 runs on all the 4 functions.** $(\max - \min)$ **shows the performance difference between the best and the worst average for the given technique.**

| $(\max - \min)$ | $Avg(all)$ | $\mathcal{F} = \sum n$ | AOS tech. |
|---|---|---|---|
| **95** | 5120±258 | 5088±252 | **AUC-B** |
| **62** | 5152± 263 | 5112± 284 | **SR-B** |
| 133 | 5199± 285 | 5139± 232 | XNor AP |
| 256 | 5298± 274 | 5173± 252 | XAbs AP |
| 105 | 5407± 448 | 5374± 471 | XNor PM |
| 144 | 5438± 436 | 5364± 354 | XAbs PM |
| **0** | 5499± 309 | 5499± 312 | **F-SR-B** |
| 230 | 5673± 420 | 5631±417 | XNor MAB |
| **0** | 6147± 458 | 6147± 462 | **F-AUC-B** |
| 2099 | 6367± 1416 | 5096± 240 | XAbs DMAB |
| 6144 | 6627± 3601 | 5090± 236 | XNor DMAB |
| 8267 | 11263±3811 | 8322±652 | XAbs MAB |

## 4.4 Results on Artificial Scenarios

A second set of experiments was done to analyze the agility of the two newly proposed AOS combinations (AUC-B and SR-B) to adapt to different situations, compared to three existent ones (DMAB, MAB and AP). The PM technique was neglected here due to its constant very low performance. Table 3 presents the complete results of such analysis on the 3 artificial scenarios described in Section 4.1, with 2 different epoch lengths $\Delta T$, 200 and 2000. Three different types of information are shown for each technique/problem/epoch length. The first line shows the best configuration found by the F-Race for the given scenario/epoch, including the *Credit Assignment* method for DMAB, MAB, and AP as discussed in Section 4.2 (A and B denote $\alpha$ and $\beta$ respectively). Note that different values of the *Credit Assignment* were found in the optimal setting, justifying that it being viewed as an hyper-parameter. The second line displays the rate of selection of the best operator, and the third line, the cumulative reward gathered during the run, the two latter measures being averaged over 50 runs. Since the main motivation for the proposal of the new AOS schemes is to have a higher robustness w.r.t. their hyper-parameters in different cases, even with some cost in terms of performance, both the unsigned Wilcoxon rank sum, and the Kolmogorov-Smirnov non-parametric tests were used to assess the performance of the new mechanisms.

The AUC-B method was able to improve over the performance of AP in 5 out of 6 cases, being significantly better in 4 of them, and equivalent in the other two. Compared

**Table 3: Empirical comparison on the Artificial Scenarios. For each of the analyzed techniques, on each problem and epoch length ($\Delta T$), the first line shows the best configuration found by the F-Race; the second line shows the rate at which the given AOS scheme was able to select the best operator; while the last line shows the achieved cumulative reward. Both empirical measures are averaged over 50 runs, with the confidence interval being also shown.**

| Problem | $\Delta T$ | AUC-B | SR-B | DMAB | MAB | AP |
|---|---|---|---|---|---|---|
| *Uniform* | 200 | C.1D1.0W50<br>79.0 ± 2.1<br>8937 ± 83 | C.1D0.5W50<br>68.9 ± 3.7 △<br>8469 ± 92 | XAbs C.5G5W10<br>91.3 ± 0.7<br>9570.63 ± 40.5384 | XNor C.5W10<br>69.8 ± 5.6 △ ○<br>8907.81 ± 141.655 | XAbs P.05A.9B.9W10<br>70.7 ± 2.7 ▲ ●<br>8625.88 ± 85.3378 |
| | 2000 | C.1D1.0W100<br>93.6 ± 0.3<br>96763 ± 149 | C.1D0.5W100<br>90.9 ± 1.1 △<br>95501 ± 234 | XAbs C.5G10W10<br>98.9 ± 0.1<br>99491.7 ± 78.6411 | XAbs C5W10<br>91.7 ± 0.2 ▲ ●<br>96657.9 ± 99.0754 | XAbs P.05A.9B.9W10<br>78.9 ± 0.4 ▲ ●<br>89560.4 ± 153.668 |
| *Boolean* | 200 | C.0001D1.0W50<br>47.6 ± 8.8<br>8055 ± 406 | C.5D0.7W50<br>36.4 ± 8.3 △<br>7477 ± 353 | AvgAbs C1G5W10<br>48.3 ± 6.4 △ ○<br>8121.6 ± 264.902 | AvgAbs C5W10<br>45.7 ± 6.2 △ ○<br>7921.4 ± 312.691 | AvgAbs P.05A.9B.6W10<br>45.0 ± 5.0 △ ○<br>7870.6 ± 271.138 |
| | 2000 | C.0005D1.0W100<br>76.4 ± 5.1<br>92172 ± 1538 | C.5D0.1W50<br>66.4 ± 5.4 △<br>87256 ± 1321 | AvgAbs C5G250W10<br>81.3 ± 3.1<br>93327.2 ± 1120.9 | AvgAbs C10W10<br>78.4 ± 2.4 △ ○<br>91478.8 ± 1036.54 | AvgNor P0A.3B.9W10<br>48.4 ± 13.6 ▲ ●<br>85802.8 ± 4244.44 |
| *Outlier* | 200 | C1D0.5W50<br>37.9 ± 6.3<br>7303 ± 679 | C1D0.5W100<br>37.5 ± 7.3 △<br>7223 ± 625 | XAbs C25G.1W50<br>39.0 ± 9.7 △ ○<br>7561.8 ± 635.201 | XAbs C50W50<br>36.8 ± 11.3 △ ○<br>7366.6 ± 582.814 | XAbs P.05A.9B.6W50<br>40.8 ± 7.2 △ ○<br>7449.2 ± 641.07 |
| | 2000 | C.5D0.7W100<br>72.8 ± 4.9<br>89210 ± 2527 | C.5D0.1W100<br>72.3 ± 6.3 △<br>89094 ± 2853 | XAbs C50G500W50<br>76.1 ± 6.0<br>91621.8 ± 2672.81 | XAbs C100W50<br>81.2 ± 2.4<br>92118.8 ± 1982.15 | XAbs P.05A.9B.1W50<br>70.6 ± 3.1 ▲ ●<br>86595 ± 2035.29 |

▲ indicates that the cumulative reward achieved by the AUC-B over 50 runs is significantly better than the one achieved by the given technique, according to the Kolmogorov-Smirnov and/or the Wilcoxon signed-rank test at $\alpha = 0.05$.
△ indicates that the same cumulative reward of AUC-B is statistically equivalent to the one achieved by the given technique.
● and ○, respectively, show the same information concerning the performance of the SR-B.

to the MAB AOS, it is significantly better or equivalent in 5 of the cases. Finally, it is statistically equivalent to the DMAB in only two of the test cases. Indeed, the DMAB method has been demonstrated to achieve very high performances [9, 10, 11], but the price to pay for this is a very expensive tuning stage (3800 configurations for the Racing, see Section 4.2) with very sensitive parameters. Concerning the SR-B approach, its performances are globally a bit worse than those of the AUC-B. However, given the high number of statistically equivalent results, and some better results found with both new techniques in these very different situations, it becomes clear that the goal of this work, i.e., improving the robustness of the bandit-based approaches while not losing too much in terms of performance, can be considered achieved.

## 5. DISCUSSION AND FURTHER WORK

One might see as a drawback the 3 hyper-parameters that still need to be tuned for these rank-based AOS schemes (the scaling factor $C$, the decay factor $D$ and the window size $W$). Although an off-line tuning technique as the F-Race might still be very helpful, given the mentioned (and expected) robustness, a much cheaper solution than that proposed in Section 4.2 can be used, insomuch that less configurations in fact need to be tried in the Racing procedure. As already discussed in Section 2.3, and mentioned in Section 4.2, hyper-parameter $C$ only has to tune the balance between exploitation and exploration. Thus only an order of magnitude seems to be needed. Furthermore, the precise value of the decay parameter $D$ does not seem to matter much, as long as it allows to distinguish between two very different situations, one with linear decay ($D = 1$) and one with relatively fast decay (e.g., $D = 0.5$). Finally, the window size $W$ can probably be limited to fewer values. And indeed, in all scenarios presented in this paper (Section 4.1), as well as in many others not presented here due to space limitation, at least one of the 16 configurations corresponding to $D \in \{0.5, 1.0\}$ ; $C \in \{0.01, 0.1, 1, 10\}$ ; and $W \in \{50, 100\}$ was found to perform statistically equivalently to the best configuration found by the F-Race.

Going even further, one interpretation of hyper-parameter $W$ is that it triggers the memory decay of the AOS mechanism: large $W$ will imply very conservative operator selection, unlikely to change rapidly, whereas short values will result in a very quick forgetting of the past. Nevertheless, its effect is tightly coupled with that of the decay $D$, that balances between favoring the more "extreme" operators (small $D$ will greatly favor the first ranked operators), and the more "average" ones ($D = 1$ will give a linear decay in the weight of the rewards in the window, so all the rewards "matter" somehow, thus favoring the ones that have been efficient often, even if generating only a small fitness improvement). Whereas small values of $W$ will tend to hide any influence of $D$, a small $D$ with a large $W$ will be similar to the Extreme-based *Credit Assignment* advocated in [9]. Those parameters should be set together, and, if available, based on some knowledge about the fitness landscape . . . which means that they should be made adaptive, as it is unlikely that the same setting can be optimal from the beginning to the end of the search. This will be the topic of further work.

Another important issue regards the fully comparison-based variants introduced in Section 3.2: their performances are worse on the OneMax problem than the best variants of the other techniques, though rather surprisingly, SR-B performs significantly better than AUC-B, and almost as good as PM. Analyzing deeply the results, and in particular looking at the frequency of use of all operators comparing it to the optimal strategy [9, 10] shows surprising instabilities in areas where one operators clearly dominates the others.

Tracking and suppressing such strange behaviors, hopefully raising the performances of those AOS algorithms, is a path to be explored.

## 6. CONCLUSION

This paper has proposed two new *Credit Assignment* methods in order to address the weaknesses of previous AOS (most particularly bandit-based AOS) regarding the robustness with respect to fitness transformations and, accordingly, hyper-parameter setting. Their output is directly used as empirical reward within a bandit-based *Operator Selection Rule*, and the resulting AOS perform achieve their promises: the performances of the AUC-B AOS are only outperformed by those of the the highly efficient but highly sensitive to hyper-parameter tuning (and to fitness transformations) DMAB, and are better than most previously proposed AOS methods, on both the OneMax problem and some artificial scenarios.

Furthermore, because they rely only on the ranks of the raw rewards awarded to operators, those original AOS are more robust than other methods with respect to their own hyper-parameters, and only a small set of configurations of those hyper-parameter need to be explored when no information is available about the fitness landscape.

Finally, they have been coupled to pure comparison-based raw reward – the rank of the newborn offspring in the current sliding window of offspring, thus building a truly comparison-based AOS. Though the performances of F-SR-B are still behind those of other AOS on the OneMax problem, their complete invariance with respect to monotonous transformations of the fitness make them a good basis for the development of future parameter-less AOS.

## 7. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[2] H. Barbosa and A. Sá. On adaptive operator probabilities in real coded genetic algorithms. In *XX Intl. Conf. of Chilean Computer Science Society*, 2000.

[3] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W. B. Langdon et al., editor, *Proc. Genetic and Evolutionary Computation Conference*, pages 11–18. Morgan Kaufmann, 2002.

[4] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997.

[5] L. Da Costa, A. Fialho, M. Schoenauer, and M. Sebag. Adaptive operator selection with dynamic multi-armed bandits. In M. Keijzer et al., editor, *Proc. GECCO*, pages 913–920. ACM, 2008.

[6] L. Davis. Adapting operator probabilities in genetic algorithms. In J. D. Schaffer, editor, *Proc. Intl. Conference on Genetic Algorithms*, pages 61–69. Morgan Kaufmann, 1989.

[7] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[8] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In Lobo, F.G. et al., editor, *Parameter Setting in Evolutionary Algorithms*, pages 19–46. Springer, 2007.

[9] A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Extreme value based adaptive operator selection. In G. Rudolph et al., editor, *Proc. PPSN X*, volume 5199 of *LNCS*, pages 175–184. Springer, 2008.

[10] A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In T. Stützle et al., editor, *Proc. Learning and Intelligent OptimizatioN*, 2009.

[11] A. Fialho, M. Schoenauer, and M. Sebag. Analysis of adaptive operator selection techniques on the royal road and long k-path problems. In G. Raidl et al., editor, *Proc. Genetic and Evolutionary Computation Conference*, pages 779–786. ACM, 2009.

[12] S. Gelly, S. Ruette, and O. Teytaud. Comparison-based algorithms are robust and randomized algorithms are anytime. *Evolutionary Computation Journal*, 15(4):411–434, 2007.

[13] D. Goldberg. Probability matching, the magnitude of reinforcement, and classifier system bidding. *Machine Learning*, 5(4):407–426, 1990.

[14] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[15] C. Hartland, S. Gelly, N. Baskiotis, O. Teytaud, and M. Sebag. Multi-armed bandit, dynamic environments and meta-bandits. In *Online Trading of Exploration and Exploitation Workshop, NIPS*, 2006.

[16] D. Hinkley. Inference about the change point from cumulative sum-tests. *Biometrika*, 58(3):509–523, 1970.

[17] B. Julstrom. What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm on genetic algorithms. In L. J. Eshelman et al., editor, *Proc. ICGA*, pages 81–87. Morgan Kaufmann, 1995.

[18] F. Lobo and D. Goldberg. Decision making in a hybrid genetic algorithm. In B. Porto, editor, *Proc. ICEC'97*, pages 121–125. IEEE, 1997.

[19] J. Maturana, A. Fialho, F. Saubion, M. Schoenauer, and M. Sebag. Extreme compass and dynamic multi-armed bandits for adaptive operator selection. In *Proc. Congress on Evolutionary Computation*, pages 365–372. IEEE, 2009.

[20] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In H.-G. Beyer, editor, *Proc. Genetic and Evolutionary Computation Conference*, pages 1539–1546. ACM, 2005.

[21] D. Thierens. Adaptive strategies for operator allocation. In Lobo, F.G. et al., editor, *Parameter Setting in Evolutionary Algorithms*, pages 77–90. Springer, 2007.

[22] A. Tuson and P. Ross. Adapting operator settings in genetic algorithms. *Evolutionary Computation*, 6(2):161–184, 1998.

[23] J. Whitacre, T. Pham, and R. Sarker. Use of statistical outlier detection method in adaptive evolutionary algorithms. In M. Keijzer, editor, *Proc. GECCO*, pages 1345–1352. ACM, 2006.