Toward Comparison-based Adaptive Operator Selection

Álvaro Fialho[1], Marc Schoenauer[1,2], Michèle Sebag[1,2]
[1]Microsoft Research–INRIA Joint Centre
Parc Orsay Université
91893 Orsay Cedex, France
[2]Project-Team TAO, INRIA Saclay - Île-de-France,
LRI (UMR CNRS 8623), Université Paris-Sud
91405 Orsay Cedex, France
FirstName.LastName@inria.fr

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 schemes make direct use of the fitness gain between parent and offspring. A first issue is that the Operator Selection technique that uses such kind of Credit Assignment is likely to be highly dependent on the a priori unknown bounds of the fitness function. Additionally, these 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, what is a usual 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, Ranks, ROC Area Under Curve, Multi-Armed Bandits

GECCO'10, July 7–11, 2010, Portland, Oregon, USA.

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 leads to Euclidian geometry.

In the realm of Evolutionary Computation, the importance of invariance has been stressed, too. Some Evolutionary Algorithms (EAs), for instance, are invariant under monotonous transformations of the fitness function, what is achieved by the use 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) [13]. The best possible consequence of invariances within the EC community is illustrated by the success of the CMA-ES algorithm [15]: it is not only comparison-based, but it is 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 [15]. 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 hyper-parameters of the AOS method. Secondly, in any case, the fitness gains generally decrease as evolution proceeds, and no static tuning can properly handle 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 these 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 hyper-parameter setting. Building on these 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 its fitness function, and on artificial scenarios for comparison with other existing AOS combinations. 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, a.k.a. *parameter control* in the literature [8, 9], and will not discuss any off-line setting, a.k.a. *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 [9, p.21] and references therein).

Amongst parameter control methods, 3 categories are usually distinguished [9]. *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". The most successful self-adaptive EA to-date are the early Evolution Strategies [2]. However, self-adaptive methods require the exploration of both the search space of the variables and that of the parameter values, and are can hence sometimes be outperformed by well-designed *adaptive* methods, that modify the parameter values based on the informations given by the search itself.

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 [12], 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 the operator to be applied based on the rewards gathered by them in the past.

2.1 Credit Assignment

Starting from Davis' seminal work [7], 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* round.

Most methods use as raw reward the fitness improvement of the newborn offspring compared to a base individual, that might be its parent [19, 24, 3, 10, 12], the current best in the population [7], or the median individual of the current population [18]. Two more recent approaches, targeted toward highly multi-modal problems, considered both fitness improvement and the variation of some diversity measure to design the reward of operators: aggregating them in a mechanism termed *Compass* [20], or treating the issue as a 2-objective problem, and using as a reward the *Pareto Dominance* score [21]. However, because this paper is concerned with robustness against fitness transformations, which has no relationship with any diversity measure, such approaches will not be considered here. Nevertheless, all results of the present paper could be applied in turn to the fitness-based part of both the above *Credit Assignment* schemes.

Regarding 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) [7, 18], based on the claim that the existence of efficient parents is indeed as important as the creation of improved offspring. However, a more recent work suggested that rewarding the ancestors' operators can sometimes degrade the performance [3].

Finally, the *Credit Assignment* transforms the raw reward into the actual credit that updates the empirical estimates of each operator, used in turn by the *Operator Selection* rule to select the operator for the next offspring generation. The existing approaches here differ in the statistics that are considered in order to compute such credit. 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 [25], 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. Reported comparative results with other *Credit Assignment* schemes demonstrate the benefit of this approach, over a set of continuous benchmark problems [25], and in the GA context [10, 12].

2.2 Operator Selection Rules

The *Operator Selection* rules usually attach an empirical quality to each operator, differing mostly on how they use such information to select the operator to be applied. Two

families of such rules are briefly reviewed in this paper (and used as baseline for comparison with the proposed methods).

The *probability-based* methods *Probability Matching* (PM) and *Adaptive Pursuit* (AP) use the empirical qualities to update operator probabilities, which are then used to select the operator to be applied by means of a roulette wheel. A user-defined relaxation factor $\alpha \in]0, 1]$ is often used to control the impact of the last received reward on the update of the empirical estimate.

PM is the most common and straightforward approach to *Operator Selection* [14, 19, 3]. As its name says, the probability p of applying each operator is proportional to its empirical quality. A minimal probability value p_{min} is usually implemented, so that none of the operators gets "lost" during the process (*i.e.*, probability 0). As all probabilities sum up to one, for k operators, the maximum probability for the selection of any operator will be $p_{max} = 1 - k * p_{min}$, which badly impacts the performance of this method [22].

AP implements a winner-takes-all strategy to partially address this drawback [22, 23]. Briefly, after each application, the probability of the best operator (*i.e.*, the one with maximum empirical quality) is moved towards p_{max}, while the other are equally decreased (towards p_{min}) so that the sum remains 1. Though it also relies on a user-defined lower bound p_{min} to maintain a minimal level of exploration, another parameter, β, is used to control the greediness of the strategy, *i.e.*, how fast the probability of selecting the current best (resp. the others) will converge to p_{max} (resp. to p_{min}). AP obtains better results than PM in most, if not all, reported works [23, 11, 12].

The *bandit-based* methods, *Multi-Armed Bandit* (MAB) and *Dynamic Multi-Armed Bandit* (DMAB) [6, 10, 11, 12], use variants of the multi-armed bandit paradigm to deterministically choose the operator to be applied. 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, one of its variants, the *Upper Confidence Bound* (UCB) strategy [1], provides guarantees of asymptotic optimality in terms of cumulated reward, being phrased as *"Optimism in front of the Unknown"*.

In the AOS context, each operator is considered as an arm, with the rewards being 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 empirical quality $\hat{q}_{i,t}$ is the average of the received rewards up to time t, and confidence bounds are computed based on the frequency of use of each arm. Next arm to pull is the one with the highest possible value within these bounds, formally defined as follows

$$\arg\max_i \left(\hat{q}_{i,t} + C \sqrt{\frac{2 \log \sum_k n_{k,t}}{n_{i,t}}} \right) \quad (1)$$

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 the empirical quality $\hat{q}$ lies in $[0, 1]$; while in the AOS framework the range of the rewards is usually unknown, thus a careful tuning of C is required [11, 12]. UCB rule with a scaling factor C is what we refer to as the MAB algorithm.

However, in the original bandits framework, the rewards distribution is static, *i.e.*, it is assumed to remain the same all along the experiment, what is obviously not the case in the AOS context. Although the UCB formula theoretically guarantees a minimal exploration of all the arms, it will asymptotically take exponential time to detect a change in the distributions. This was the motivation that led to the design of the *Dynamic Multi-Armed Bandit* (DMAB) algorithm [16], that incorporates the Page-Hinkley change detection statistical test [17] into the original MAB algorithm: when this test triggers (according to a user-defined parameter γ), the Multi-Armed Bandit process is restarted from scratch.

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.

Firstly, in the mentioned references for each technique, the fitness improvement of the newborn offspring with respect to its parent was used (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.*, $\mathcal{F} \to \mathcal{F} + a$ 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 algorithms on the other side. Both PM and AP are invariant by linear scaling of the fitness function (*i.e.*, $\mathcal{F} \to a \times \mathcal{F}$ 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 α – though in practice the same settings are generally used. Experimental results presented in Section 4.3 will shed more light on this issue.

On the other hand, for all bandit-based AOS mechanisms [6, 10], the actual value of the fitness gains are directly used in the UCB formula (Eq. 1). 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 γ 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 [12], but did not seem to much 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 hyper-parameter configuration for the bandit-based AOSs.

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

3. ROBUST CREDIT ASSIGNMENTS

In this Section, firstly, two original *Credit Assignment* schemes are proposed to address the issues about robustness with respect to fitness transformations listed above,

using only the ranks of the raw rewards measuring the recent impacts of the operators, *i.e.*, the fitness improvements. In order to achieve 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 the parameter C in the UCB formula (Eq. 1), as 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 over their ranks.

A sliding window of size W is used to store the impacts of the recent W applications of operators. Each slot in this window contains the index of the operator that has been applied and the corresponding fitness improvement.

Two approaches are proposed to assign credit from the ranked impact measures, both sharing the same ranking assignment scheme, as follows. Each position in the window is ranked, with the position r initially receiving a rank-value of $(W - r)$. A decay factor $D \in]0, 1]$ is then applied over these rank-values, so that the top-ranked rewards exert a higher influence on the calculation of the credit assigned to each operator (following, and somehow smoothening the intuition of the extreme value based rewards [10]). The final rank value corresponding to each operator application is then calculated as $D^r(W - r)$. Parameter D thus defines how skewed the ranking distribution is ($D = 1$ represents the linear decay). The smaller D, the faster the decay.

The **Sum of Ranks** (SR) method, as its name already says, credits the operators with the sum of the ranks of the rewards given after its applications, normalized by the sum of all the rank-values, so that the sum of the credits assigned to all operators sum up to 1 . Being K the number of operators, the operator i is rewarded at time t as follows:

$$SR_{i,t} = \frac{\sum_{op_r=i} D^r(W - r)}{\sum_{r=1}^{W} D^r(W - r)} \quad (2)$$

The **AUC** method borrows ideas from the *Area Under the Curve*, a criterion used in Machine Leaning to compare binary classifications rules [5]. Computing the reward of a given operator amounts to going down the sorted list, and drawing, starting from the origin, the *Receiving Operator Curve* (ROC) by adding a vertical segment each time the operator under scrutiny is found in the list, a horizontal one otherwise, and a diagonal in case of ties.

Algorithm 1 describes more formally how to calculate the AUC reward, while Figure 1 illustrates this procedure on a small example without decay, for the sake of clarity. 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 $D^r(W - r)$, with the dotted lines being moved accordingly. The AUC credit is represented by the grey area, lately normalized by the sum of the AUCs assigned to each operator, so that their sum equals to 1.

Because both quantities SR and AUC are already some kind of statistics over the time window W, they can be used directly in the UCB formula, replacing $\hat{q}$ in Eq. 1. The resulting AOS mechanisms are called respectively *Sum of Ranks-Bandit* (SR-B) and *AUC-Bandit* (AUC-B). Note that

Algorithm 1 AUC calculation

```
1: x ← y ← area ← 0
2: for r ← 1 to W do
3:    Δ_r ← D^r(W − r)
4:    tiesX ← countTiesTargetOp();
5:    tiesY ← countTiesOtherOps()
6:    if tiesX||tiesY then                    // ties: diagonal trace
7:       Δ_tie ← 0
8:       for s ← r to (r + tiesX + tiesY) do
9:          Δ_tie ← Δ_tie + (D^s(W − s))/(tiesX + tiesY)
10:      x ← x + tiesX ∗ Δ_tie
11:      area ← area + y ∗ Δ_tie ∗ tiesX        // rectangle below
12:      y ← y + tiesY ∗ Δ_tie
13:      area ← area + .5 ∗ Δ²_tie ∗ tiesX ∗ tiesY      // triangle
14:      r ← r + tiesX + tiesY
15:   else if op_r == op_target then          // target op.: vertical
16:      y ← y + Δ_r
17:   else                                    // other operators: horizontal
18:      x ← x + Δ_r
19:      area ← area + y ∗ Δ_r
20: return area
```

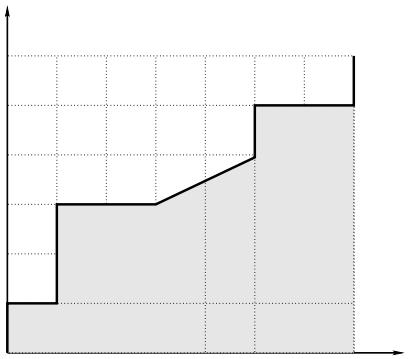

Figure 1: Sample computation of AUC reward: only two operators are involved, and the sorted list contains the operators in the order (1 2 1 1 2 2 [2 2 1] 1 2 2 1), with [2 2 1] meaning that these 3 positions have the same raw reward, leading to the diagonal line between points (3 3) and (5 4) (dotted lines are spaced by 1). In case of decay, the width of the squares would decrease leftward and upward.

a few trials using SR or AUC as a reward to be fed in one of the *Operator Selection* techniques described in Section 2.2 performed rather poorly, two layers of statistics somehow diluting the interesting characteristics of the rewards.

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, as the raw rewards that are used here are actual values of fitness improvements, some monotonous transformations 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 application of the operators. Replacing the fitness improvements by the raw fitnesses of the newborn offspring, these *Credit Assignment* schemes allow us to compute a reward that is fully comparison-based, as only sorting some

fitness values is required. Just the fitnesses of the offspring that improved over their parents are considered, a null reward is assigned otherwise.

These two AOS combinations are 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 pre-defined reward, and not some fitness improvement. 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) will be distinguished.

4. EXPERIMENTAL RESULTS

This Section details the experimental validation of the proposed rank-based bandit approaches. As it is a motivation of the present work, their behavior with respect to transformation of the fitness will be compared to that of previously proposed AOS combinations, on the well-controlled experimental benchmark offered by the OneMax problem [10]. Then, more extensive comparisons will be performed on artificial scenarios, as proposed in [23, 6].

4.1 Scenarios

The well-known **OneMax problem**, a.k.a. the "drosophila of EC", has already been used to assess the performance of probability-based and bandit-based AOS [10, 11]. 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 the 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, as the optimal strategy for operator choice is completely known [10].

Artificial Scenarios are another popular setting to empirically validate AOS schemes, initiated with Thierens' original benchmark [22]. A set of 5 "operators" have a prescribed reward distribution, that varies along epochs of ΔT 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[2]. The performance associated to an AOS 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 AOS with no information whatsoever. For this reason, two variants of the *Uniform* benchmark, respectively referred to as *Boolean* and *Outlier*, have been proposed in [6].

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 AOS 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 AOS 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},Avg{Abs,Nor}	Credit Assign. type
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	α	$\{.1, .3, .6, .9\}$	Adaptation rate
AP	β	$\{.1, .3, .6, .9\}$	Learning rate
All bandit	$\mathcal{C}$	$\{\{1, 5\}.10^{\{-4 \leq i \leq 2\}}\}$	Scaling factor
DMAB(PH)	γ	$Range(\mathcal{C})$, $\{250, 500, 1000\}$	PH threshold

Table 1: AOS Hyper-parameters and value range

As previously mentioned, all the *Credit Assignment* and *Operator Selection* schemes have 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 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 [12]. The basic idea of Racing, introduced in the field of Evolutionary Computation in [4], is to start running a 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 [4], the elimination of unpromising configurations starts after 11 runs, and the racing is stopped as soon as a single configuration remains, or after 50 runs of the remain-

[2]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.

ing configurations have been performed. The criterion used for its statistical test 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. Firstly, the first parameter in this table was not yet mentioned in this paper: the *Credit Assignment* type for all methods from earlier work (PM, AP, MAB, and DMAB). As in [12], the actual reward that the AOS computes from the raw reward (the fitness improvement in the OneMax scenario, the predefined 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 a 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 tested types of *Credit Assignment*, the number of different candidate parameter configurations that were tried in the Racing for the different techniques ranges from 4×64 for PM and 4×256 for AP to 4×56 for MAB and 4×952 for DMAB, due to the high uncertainty (and high sensitivity) of the Page Hinkley parameter γ. For the newly proposed techniques, 'only' 336 configurations were tried, given the initial lack of knowledge about their behavior; however, trying just 16 configurations would be enough to achieve the same level of performance, as discussed in Section 5.

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 same configuration was used on different transformations of the OneMax fitness $\mathcal{F}$, namely $\log(\mathcal{F})$, $\exp(\mathcal{F})$, and $\mathcal{F}^2$.

The complete results for the first scenario (two operators, 1-Bit and 5-Bit mutation) can be found on Table 2. The second scenario is suppressed here, as it raises basically the same conclusions than the presented. Also, the results for the AOSs using the *Average*-based *Credit Assignment* are not shown, because in this case they are not competitive (though presenting a smaller variance than the *Extreme* in the results w.r.t. the presented fitness transformations).

These 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. As OneMax is a very easy problem, AP 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 for $\exp(\mathcal{F})$ sets $W = 100$ and $\beta = 0.9$: as 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.

As expected, the newly proposed methods based on the rank of the fitness improvements, AUC-B and SR-B, achieved an overall better performance, while showing to be robust w.r.t. the monotonous transformations, what was not true for the previously published bandit-based approaches. Although the comparison-based counterparts F-AUC-B and F-SR-B showed to be less competitive, their invariance characteristic was verified, and might even be more beneficial in a bigger and more difficult class of problems.

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

$(\max - \min)$	$Avg(all)$	$\mathcal{F} = \sum n$	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 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 ΔT, 200 and 2000. 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 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 [10, 11, 12], but the price to pay for this is a very expensive tuning stage (summing up to 3808 configurations for the Racing, see Section 4.2) with very sensitive parameters.

Table 3: Empirical comparison on the Artificial Scenarios. For each of the analyzed techniques, on each problem and epoch length (ΔT), 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	ΔT	AUC-B	SR-B	DMAB	MAB	AP
Uniform	200	C.1D1.0W50 79.0 ± 2.1 8937 ± 83	C.1D0.5W50 68.9 ± 3.7 △ 8469 ± 92	XAbs C.5G5W10 91.3 ± 0.7 9570.63 ± 40.5384	XNor C.5W10 69.8 ± 5.6 △ ○ 8907.81 ± 141.655	XAbs P.05α.9β.9W10 70.7 ± 2.7 ▲ ● 8625.88 ± 85.3378
	2000	C.1D1.0W100 93.6 ± 0.3 96763 ± 149	C.1D0.5W100 90.9 ± 1.1 △ 95501 ± 234	XAbs C.5G10W10 98.9 ± 0.1 99491.7 ± 78.6411	XAbs C5W10 91.7 ± 0.2 ▲ ● 96657.9 ± 99.0754	XAbs P.05α.9β.9W10 78.9 ± 0.4 ▲ ● 89560.4 ± 153.668
Boolean	200	C.0001D1.0W50 47.6 ± 8.8 8055 ± 406	C.5D0.7W50 36.4 ± 8.3 △ 7477 ± 353	AvgAbs C1G5W10 48.3 ± 6.4 △ ○ 8121.6 ± 264.902	AvgAbs C5W10 45.7 ± 6.2 △ ○ 7921.4 ± 312.691	AvgAbs P.05α.9β.6W10 45.0 ± 5.0 △ ○ 7870.6 ± 271.138
	2000	C.0005D1.0W100 76.4 ± 5.1 92172 ± 1538	C.5D0.1W50 66.4 ± 5.4 △ 87256 ± 1321	AvgAbs C5G250W10 81.3 ± 3.1 93327.2 ± 1120.9	AvgAbs C10W10 78.4 ± 2.4 △ ○ 91478.8 ± 1036.54	AvgNor P0α.3β.9W10 48.4 ± 13.6 ▲ ● 85802.8 ± 4244.44
Outlier	200	C1D0.5W50 37.9 ± 6.3 7303 ± 679	C1D0.5W100 37.5 ± 7.3 △ 7223 ± 625	XAbs C25G.1W50 39.0 ± 9.7 △ ○ 7561.8 ± 635.201	XAbs C50W50 36.8 ± 11.3 △ ○ 7366.6 ± 582.814	XAbs P.05α.9β.6W50 40.8 ± 7.2 △ ○ 7449.2 ± 641.07
	2000	C.5D0.7W100 72.8 ± 4.9 89210 ± 2527	C.5D0.1W100 72.3 ± 6.3 △ 89094 ± 2853	XAbs C50G500W50 76.1 ± 6.0 91621.8 ± 2672.81	XAbs C100W50 81.2 ± 2.4 92118.8 ± 1982.15	XAbs P.05α.9β.1W50 70.6 ± 3.1 ▲ ● 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.

Concerning the SR-B approach, its performances are globally a bit worse than these 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 newly proposed rank-based AOS schemes (the scaling factor C, the decay factor D and the window size W).

But, firstly, given their higher robustness w.r.t. the fitness transformations, one hyper-parameter configuration found to be the best for a given problem will also perform well in the whole class of problems defined by monotonous transformations over the original function. Additionally, as already discussed in Section 2.3, 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 over the complete set of configurations, as defined in Section 4.2.

Going even further, one interpretation of hyper-parameter W is that it triggers the memory decay of the AOS mechanism: large W will imply in 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 more often, even if generating smaller rewards). 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 [10]. These 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, F-SR-B performs significantly better than F-AUC-B, and almost as good as PM. Analyzing deeply the results of the F-AUC-B, and in particular looking at the frequency of use of all operators comparing it to the optimal strategy [10, 11] shows surprising instabilities in areas where one operators clearly dominates the others. Tracking and suppressing such strange behaviors, hopefully raising the performances of these AOS algorithms, is a path to be explored.

6. CONCLUSION

This paper has proposed two new *Credit Assignment* methods, SR and AUC, addressing some weaknesses of previous AOS (most particularly bandit-based AOS) regarding their robustness with respect to fitness transformations and, accordingly, their sensitivity to the values of their hyper-parameters. SR and AUC outputs are directly used as the empirical quality within a bandit-based *Operator Selection*, and the resulting AOS combinations achieve their promises: the performances of AUC-B and SR-B are only outperformed by these of 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 analyzed scenarios.

Furthermore, because they rely only on the ranks of the raw rewards awarded to operators, these original AOSs are more robust than other methods w.r.t. the tuning of their own hyper-parameters. Hence, only a small set of configurations need to be explored when no information is available about the fitness landscape. Indeed, due to this improved robustness, the same hyper-parameter configuration will perform well on a whole class of problems defined by monotonous transformations.

Finally, SR and AUC have also been coupled to a pure comparison-based raw reward, the rank of the newborn offspring in the current sliding window, resulting in totally comparison-based AOS. Though their performances are still behind those of other AOSs, their complete invariance with respect to monotonous transformations of the fitness make them a good basis for the development of future AOS that will be both efficient and parameter-less.

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] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

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

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

[5] 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.

[6] 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.

[7] L. Davis. Adapting operator probabilities in genetic algorithms. In J. D. Schaffer, editor, *Proc. ICGA*, pages 61–69. Morgan Kaufmann, 1989.

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

[9] 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.

[10] 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.

[11] 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. LION 3*, pages 176–190. Springer, 2009.

[12] 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. GECCO*, pages 779–786. ACM, 2009.

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

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

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

[16] 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.

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

[18] 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.

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

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

[21] J. Maturana, F. Lardeux, and F. Saubion. Autonomous operator management for evolutionary algorithms. *Journal of Heuristics*, 2010.

[22] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In H.-G. Beyer, editor, *Proc. GECCO*, pages 1539–1546. ACM, 2005.

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

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

[25] 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.