Dynamic Multi-Armed Bandits and Extreme Value-Based Rewards for Adaptive Operator Selection in Evolutionary Algorithms

Álvaro Fialho[1], Luis Da Costa[2], Marc Schoenauer[1,2], and Michèle Sebag[1,2]

[1] Microsoft Research – INRIA Joint Centre, Orsay, France
[2] TAO team, INRIA Saclay – Île-de-France & LRI (UMR CNRS 8623), Orsay, France
`FirstName.LastName@inria.fr`

Abstract. The performance of many efficient algorithms critically depends on the tuning of their parameters, which on turn depends on the problem at hand. For example, the performance of Evolutionary Algorithms critically depends on the judicious setting of the operator rates. The Adaptive Operator Selection (AOS) heuristic that is proposed here rewards each operator based on the extreme value of the fitness improvement lately incurred by this operator, and uses a Multi-Armed Bandit (MAB) selection process based on those rewards to choose which operator to apply next. This Extreme-based Multi-Armed Bandit approach is experimentally validated against the Average-based MAB method, and is shown to outperform previously published methods, whether using a classical Average-based rewarding technique or the same Extreme-based mechanism. The validation test suite includes the easy One-Max problem and a family of hard problems known as "Long k-paths".

1 Introduction

Evolutionary Algorithms (EAs), remotely inspired from the Darwinian "survival of the fittest" principle, have been demonstrated to be efficient in tackling ill-posed optimization problems. Given a search space X, an objective function defined on X, referred to as *fitness*, and a set of elements in X, termed *population* of *individuals*, EAs iteratively proceed by (i) selecting some individuals, favoring those with better fitness; (ii) perturbing these individuals through some *variation operators*, thus generating *offspring*; (iii) evaluating the offspring fitness; (iv) replacing some individuals by some offspring, again favoring fitter offspring.

EAs have demonstrated their ability to address a wide range of optimization problems beyond the reach of standard methods, e.g. involving structured and mixed search spaces; irregular, noisy, rugged or highly constrained fitness functions. Their performance actually relies on tuning quite a few parameters (such as the population size, types of variation operators and respective application rates, types of selection mechanisms and other intrinsic parameters) depending on the problem at hand. This wealth of tunable parameters is the main reason

why EAs are still far away from being part of the standard optimization tool-boxes. Although knowledgeable users can benefit from this flexibility and take the most out of the evolutionary approach, the naive user will generally fail to appropriately tune an EA in a reasonable amount of time. Tuning the parameters to efficiently solve the problem at hand corresponds to an optimization problem *per se*, as noted in the very early days of the field [1]. Therefore, a mandatory step for EAs to "cross the chasm" and make it out of the research labs is to offer some automatic parameter setting capabilities. Accordingly, *Parameter Setting in EAs* was and still is one of the most active research topics in Evolutionary Computation [2] (section 2).

This paper specifically focuses on the control of the variation operators. Different operators play different roles in the search process, the importance of which depends on the current state: for instance, crossover operators ensure the *exploration* of wide regions of the search space in the early stages of evolution; meanwhile, mutation operators both ensure the *exploitation* and local search around the current best individuals, at any stage, and prevent the loss of diversity in the last stages of evolution. The so-called *Exploration vs Exploitation* trade-off thus relies on the mutation and crossover rates; in practice, these are most often defined by the user once for all, depending on his experience and intuition, although the need for exploration and exploitation clearly varies as the search goes on.

Adaptive Operator Selection (AOS) is meant to adaptively update the operator rates online, depending on e.g. the fitness improvement brought by the offspring. Since Davis' seminal work [3] (section 2), AOS proceeds by combining two main ingredients, illustrated in Fig. 1: The *Credit Assignment* mechanism associates a reward to each operator, reflecting the operator impact on the progress of the search (e.g. fitness improvement); the *Operator Selection* mechanism actually chooses one operator depending on the past associated rewards of all operators.

This paper investigates the combination of an *Operator Selection* and *Credit Assignment* heuristics, first described in [4] and [5] respectively. The proposed *Operator Selection* rule ([4], section 3.1), performs the dynamic operator selection by combining a Multi-Armed Bandit algorithm [6] with a statistical test for change point detection, the Page-Hinkley test [7], assuming an unbiased Credit Assignment mechanism. The proposed *Credit Assignment* ([5], section 3.2) considers the extreme values of the fitness improvement due to an operator, claiming that rare but highly beneficial "jumps" matter as much or more than frequent but small improvements.

A proof of principle of the AOS combining the above Extreme-Value-based *Credit Assignment* and the Dynamic Multi-Armed Bandit *Operator Selection* rule, referred to as *Ex-DMAB*, is presented in this paper; this proof of principle considers the easy One-Max problem and a family of hard problems, the Long *k*-paths [8]. Not only are the Long *k*-path landscapes more difficult than the One-Max ones (the former involves a single, exponentially long path leading to the global optimum, together with many shortcuts, while the latter involves many paths with linear length leading to the global optimum); overall, they
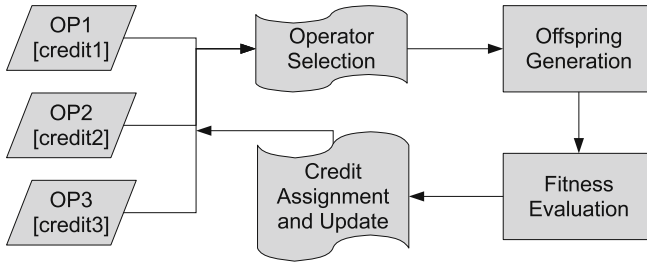
Fig. 1. General scheme of the Adaptive Operator Selection framework

can be considered to be deceptive in terms of operator selection (more on this in section 4.3). Section 4 reports on the experimental results of the approach, showing significant improvements with respect to baseline approaches. The paper concludes with some perspectives for further research.

2   Related Work

This section briefly describes and discusses the state of the art in Evolutionary Parameter Setting, focusing on *Adaptive Operator Selection*.

2.1   Parameter Setting in EAs

After [2,9,10], Parameter Setting in EAs includes two main categories of heuristics, respectively referred to as *Parameter Tuning* and *Parameter Control*:

– In *Parameter Tuning* (also known as *off-line* or *external* tuning), parameters are *tuned* before the run. This category mostly includes statistical methods derived from *Design Of Experiments* (see *e.g.*, [11,12,13,14]). Although more efficient than standard ANOVA methods, Parameter Tuning relies on extensive, computationally expensive, experiments. Furthermore, there is strong empirical evidence that the optimal parameter values actually vary between the beginning and the end of evolution; choosing the parameter values once for all thus results in a sub-optimal setting.
– In *Parameter Control* (also known as *on-line* or *internal* control), parameters are *controlled* during the run. This category can be further divided into three types of approaches:
  1. In *Deterministic Control*, parameter values are predefined functions of time, which clearly raises the question of how to define such functions (defining these *a priori* is but another Parameter Setting issue).
  2. In *Self-Adaptive Control*, parameters are encoded in the genotype, and therefore tuned and optimized "for free" by evolution itself. The main weakness of self-adaptive control is to aggregate the solution and the parameter spaces, thus increasing the overall complexity of the optimization problem.

3. In *Adaptive* (or *Feedback-Based*) *Control*, parameter values are predefined functions of the whole history of the run. Adaptive control has met significant successes in the last decade, specifically in the continuous optimization framework (see [15] and references therein).

Focusing on *Adaptive Operator Selection* (AOS), the history of the run is used to adjust the operator rates through two modules: a *Credit Assignment* module computes the operator reward based on its impact on the search progress; an *Operator Selection* module exploits these rewards and selects the operator to be applied next.

2.2 Credit Assignment

Several *Credit Assignment* mechanisms have been proposed in the literature, starting back in the late 80s with the seminal work of Davis [3]. In most approaches, the operator credit, aka *reward*, reflects the fitness of the offspring built by the operator. More specifically, the reward measures the fitness improvement over some reference fitness: that of the offspring parents [16,17,18], of the current best [3] or median [19] individual. Offspring which do not improve on the reference fitness are simply not taken into account.

In some cases however, fitness improvement is but one element relevant to the progress of evolution. Typically in multi-modal search landscapes, population diversity is equally important; it must mandatorily be preserved in order to avoid premature convergence to local optima. Based on this remark, the so-called *Compass* credit assignment [20] measures the operator ability to produce more fit individuals while preserving the population diversity.

While in all above approaches the operator reward is based on the current fitness improvement, or on the fitness improvement averaged over the last n offspring, another approach is proposed in [21]. This latter approach uses a statistical measure aimed at outlier detection, and the authors report significant improvement comparatively to other *Credit Assignment* on a set of continuous benchmark problems.

A last issue concerns the offspring contribution to the operator rewards. Most authors only reward the operator used to produce the current offspring [16,17,18]; other authors consider that it is only fair to reward the operators used to produce the offspring ancestors, e.g. using a bucket brigade algorithm [3,19].

2.3 Operator Selection Rules

The simplest and most widely used *Operator Selection* is *Probability Matching* (PM) [16,18,22]. PM implements a roulette wheel-like selection process, where the operator rate is proportional to its reward. Some care is however exercised in order to enforce a sufficient amount of exploration, through keeping the operator rate above some threshold p_{min}. Otherwise, an operator which is inefficient in the early stages of evolution would never be considered again, even though it might become the best operator later on. A side effect however is to keep the

best operator rate below $p_{max} = 1 - (K - 1)p_{min}$ being K the number of opera-
tors. In practice, all mildly relevant operators keep being selected, slowing down
evolution [23].

This drawback is partly addressed by *Adaptive Pursuit* (AP) [23], a method
originally proposed for learning automata, which implements a winner-takes-
all strategy. The main difference compared to PM is that the rate of the best
rewarded operator goes to p_{max} whereas all the others go to p_{min}; an additional
β parameter controls the greediness of the winner-take-all update.

Others, such as APGAIN [24], use a sequence of exploration/exploitation
phases. During each exploration phase, operators are uniformly selected and
their rewards are estimated; during the following exploitation phase, operators
are selected according to their reward. The fraction of generations devoted to
exploration phases (circa 25 % in [24]) is meant to catch up with the changes in
the reward distribution; unfortunately, it severely harms the population and the
progress of evolution whenever disruptive operators are considered [20].

3   Extreme Dynamic Multi-Armed Bandit

The Extreme Dynamic Multi-Armed Bandit (*Ex-DMAB*) AOS combines
Dynamic-Multi Armed Bandit as *Operator Selection* rule and Extreme Value
Based *Credit Assignment*. For the sake of self-containedness, this section sum-
marizes both heuristics, referring the interested reader respectively to [4] and [5]
for more details.

3.1   Dynamic Multi-Armed Bandit

The choice of an operator within an Evolutionary Algorithm can be viewed
as yet another instance of the *Exploration vs. Exploitation* (EvE) dilemma: on
the one hand, one wishes to select the operator with best empirical behavior
(exploitation); on the other hand, other operators should also be selected in
order to check whether the best empirical operator so far truly is the best one
(exploration). This dilemma has been intensively studied in the context of *Game
Theory* within the so-called Multi-Armed Bandit (MAB) framework [6,25].

The MAB framework involves a set of N arms; the i-th arm, when selected,
gets reward 1 with probability p_i and 0 otherwise. A MAB algorithm is a de-
cision making algorithm, selecting an arm at every time step with the goal of
maximizing the cumulative reward gathered along time. The widely studied Up-
per Confidence Bound (UCB) algorithm devised by Auer *et al.* [6], provably
maximizing the cumulative reward with optimal convergence rate, proceeds as
follows. Let $n_{i,t}$ denote the number of times the ith arm has been played up to
time t, and let $\hat{p}_{i,t}$ denote the average empirical reward received from arm i.
UCB1 selects in each time step t the arm maximizing the following quantity:

$$\hat{p}_{j,t} + C * \sqrt{\frac{\log \sum_k n_{k,t}}{n_{j,t}}} \qquad (1)$$

The left term in Eq. (1) favors the option with best average empirical reward (exploitation). The right term ensures that each arm is selected infinitely often (exploration); the lapse of time between two selections of under-optimal arms however increases exponentially. The scaling factor C controls the exploration/exploitation trade-off.

The operator selection problem can indeed be formalized as a MAB problem, taking each operator as an arm [4], with two caveats. Firstly, arms are assumed to be independent, which is definitely not the case in AOS as operators apply on the same population. Secondly, and even more importantly, the reward probabilities are fixed in standard MAB settings, whereas the operator rewards depend on the current population and the evolution stage. In other words, AOS corresponds to a dynamic MAB problem. It must be emphasized that although UCB keeps exploring all arms, it would need quite some time to detect that the best operator has changed. Therefore, a statistical change detection test was coupled with UCB in [4], defining the Dynamic MAB (DMAB) algorithm. Specifically, the Page-Hinkley (PH) test [7] is used to detect whether the empirical rewards collected for the best current operator undergo an abrupt change. Upon triggering the PH test (suggesting that the current best operator is no longer the best one), the MAB algorithm is restarted from scratch.

Formally, the PH test considers $\bar{r}_t$, the empirical average of the instant rewards $r_1, \ldots r_t$. Let e_t denote the difference $r_t - \bar{r}_t + \delta$, where δ is a tolerance parameter, and let m_t be the sum of e_i for $i = 1$ to t. The PH test is triggered when the difference between the maximum of $|m_i|$ for $i = 1$ to t, and the current $|m_t|$ is greater than a user-specified threshold γ. The PH test is thus controlled from two parameters, γ governing the trade-off between false alarms and unnoticed changes, and δ enforcing the test robustness when dealing with slowly varying distributions. Following initial experiments [4], δ is set to 0.15 in all experiments in this paper.

3.2 Extreme Value Based Credit Assignment

The second AOS component measures the operator impact on the progress of evolution 2.2. Letting $\mathcal{F}$, o and x respectively denote the fitness function (to be maximized), a variation operator and an element of the current population, the standard instant reward is set to the current fitness improvement of the offspring $(\mathcal{F}(o(x)) - \mathcal{F}(x))^+$ (the $^+$ superscript indicates the positive part of the fitness difference).

The main originality of the *Credit Assignment* proposed in [5] is to consider the *extreme* as opposed to the *average* instant reward. Let us compare an operator bringing frequent small improvements, and an operator bringing rare but large improvements. Even though both operators might have the same expected impact on evolution, with high probability an average-reward based AOS would only consider the former one: after the first trials, the former operator dominates the latter one, which is thus hardly selected thereafter, and thus prevented from gathering any further rewards. In other words, average reward-based AOS is risk-adverse. Another strategy, first investigated by [21], thus is to consider

extreme rewards. Notably, the role of extreme events in design has long been acknowledged in numerical engineering (e.g. taking into account rogue waves when dimensioning an oil rig); it receives an ever growing attention in the domain of complex systems, as extreme events govern diffusion-based processes ranging from epidemic propagation to financial markets.

The *Extreme Value Based* (EVB) *Credit Assignment* first presented in [5] proceeds as follows. To each operator o is associated a register storing the last W (positive) instant rewards collected by o. The operator reward used within the DMAB *Operator Selection* is the maximum instant reward in the operator register. This *Credit Assignment* mechanism thus involves the window size W as single parameter. W is meant to reflect the time scale of the process; if too large, operators will be applied after their optimal epoch and the switch from the previous best operator to the new best one will be delayed. If W is too small, operators causing large but infrequent jumps will be ignored (as successful events will not be observed at all in the first place) or too rapidly forgotten.

4   Experimental Results

This section reports on the empirical validation of the Extreme - Dynamic Multi-Armed Bandit (*Ex-DMAB*) AOS, combining Extreme-Value-Based *Credit Assignment* and DMAB *Operator Selection*, first described in [5].

Previous results on the One-Max problem, the "Drosophila of EC", are recalled in section 4.2, and comparatively discussed with respect to some new results obtained with Average-Value-Based *Credit Assignment*. A different family of problems, the Long k-paths [26], is considered in section 4.3. Both sets of experiments have been conducted using the same experimental setting, described in section 4.1.

4.1   Experimental Setting

All experiments consider a standard $(1 + \lambda)$-EA, where λ offspring are created from a single parent, and the best individual among the current offspring and parent becomes the parent in the next generation. For the sake of reproducibility, the initial individual is set to $(0, \dots, 0)$.

For the simplicity of assessment, the AOS only considers mutation operators: the standard $1/\ell$ bit-flip operator (every bit is flipped with probability $1/\ell$, where ℓ is the bit-string length), the 1-bit, 3-bit and 5-bit mutation operators (the b-bit mutation flips exactly b bits, uniformly selected in the parent). This setting makes it feasible to compute the optimal mutation operator depending on the stage of evolution (fitness of the current parent), using Monte-Carlo simulations. Two baseline approaches are considered: the first one uniformly selects an operator out of the whole set of operators; the second one selects an operator out of the best two operators.

Two *Credit Assignment* procedures are considered: the Extreme-Value and the Average-Value based reward (out of the last W instant rewards for the operator).

These are combined with three *Operator Selection*s: AP, PM and DMAB. The results obtained with PM will be omitted as this *Operator Selection* is found significantly dominated by the other two.

Every AOS is assessed from the average time-to-solution, averaged over 50 independent runs. The ability of each AOS to correctly identify the best operator is also considered.

The best parameters of every considered AOS have been computed offline, in order to compare them at their best level of performance (see [4,5]). For the One-Max scenario, the parameters were determined after a Design of Experiment campaign [4,5]. For the Long k-path, the following set of values was tried for each parameter: for AP and PM, $p_{min} \in \{0, .05, .1, .2\}$; $\alpha|\beta \in \{.1, .3, .6, .9\}$; for DMAB, $C \in \{.1, .5, 1, 5, 10, 50, 100\}$; $\gamma \in \{1, 5, 10, 25, 50, 100, 250, 500, 1000\}$; and for all techniques, concerning the *Credit Assignment*, $W \in \{50, 500\}$. Given the number of possible configurations, the F-Race [11] method was used. Formally, racing techniques proceed by pruning every configuration as soon as it is not going to be the best one after the available statistical evidence. The F-Race was applied using a confidence level of 95%, with 11 runs being done for each configuration before the first elimination round, up to 50 runs done or a single candidate configuration left.

4.2   The One-Max Problem

The One Max problem involves an unimodal fitness function that simply counts the number of "1"s in the individual binary bitstring. The only difficulty comes from the size of the problem; in the presented experiments, the size N of the bitstring is 10,000. This problem is viewed as a "sterile EC-like" environment, where the ideal AOS behavior can be computed. Fig. 2 (bottom) displays the optimal mutation operators for a $(1 + 50)$-EA, depending on the stage of evolution; for each fitness of the current parent, the expected fitness improvement (estimated over 100 independent runs) of a $(1+50)$-EA is computed. The landscape presented in such figure thus serves as a reference to assess the basic skills of an AOS mechanism: the ability to pick up the best operator and stick to it as long as appropriate, to catch up the changes and switch to the next best operator in transition phases, and to remain efficient in desert phases.

Fig. 2 displays the operator rates of *Ex-DMAB* and *Ex-AP* (averaged over 50 runs) against the "oracle"; the vertical grey lines indicate the changes of the current best mutation operator of the oracle.

Table 1 summarizes the performance of all approaches, together with their optimal setting. The Extreme Value-Based *Credit Assignment*, coupled with either AP or DMAB, closely matches the optimal behavior and significantly improves on the baseline approaches (see [5] for more detail). Interestingly, the Extreme Value-Based *Credit Assignment* appears to be more stable than the Average Value-Based one (the performance of the latter is much degraded when combined with DMAB), despite the smoothness of the One-Max landscape (which should thus make little difference between average and extreme rewards).
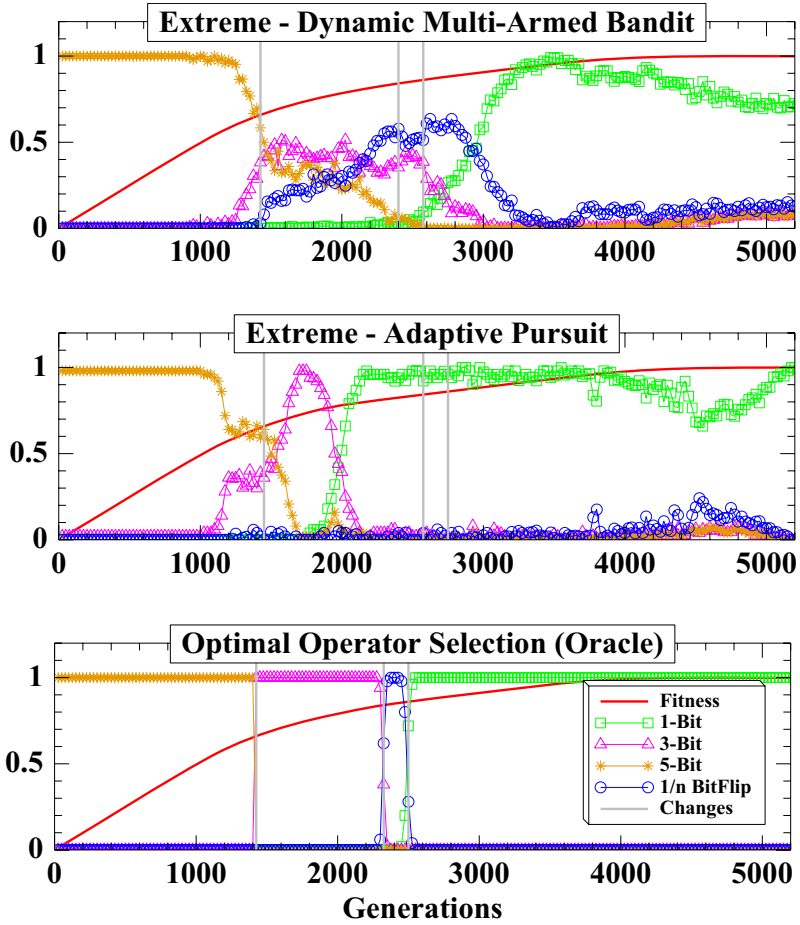
Fig. 2. The *Ex-DMAB* (top) and the *Ex-AP* (middle) AOS compared with the Optimal Operator Selection (bottom) on the 10,000 bit One-Max: operator selection rates averaged on 50 runs

Table 1. Comparative AOS Results on the 10,000 One-Max problem, averaged over 50 runs ($W = 50$)

Credit Assignment	Operator Selection	Configuration	Gens. to Optimum
Extreme	DMAB	$C = 1; \gamma = 250$	**5467** $\pm$ 513
Average		$C = 10; \gamma = 25$	7727 $\pm$ 642
Extreme	Adaptive Pursuit	$p_{min} = 0; \alpha = .3; \beta = .3$	**5478** $\pm$ 299
Average		$p_{min} = .05; \alpha = .1; \beta = .9$	5830 $\pm$ 324
-	Optimal Strategy	Given by "Oracle"	**5069** $\pm$ 292
-	Best Naive	$\mathcal{U}(1\text{-Bit}+5\text{-Bit})$	6793 $\pm$ 625
-	Complete Naive	$\mathcal{U}(4 \text{ ops.})$	7813 $\pm$ 708

4.3   The Long k-Path Problem

First proposed by [26], Long Paths are unimodal problems designed to challenge local search algorithms. Specifically, the optimum can be found by following a path in the fitness landscape, the length of which increases exponentially w.r.t. the bitstring length ℓ. Solving the Long Path using the 1-bit mutation thus requires a time increasing exponentially with ℓ.

A generalization of Long Path problems was proposed by [8], referred to as Long k-path, where k is the minimal number of bits to be simultaneously flipped in order to take a shortcut on the path. Long k-path problems have the following properties [27]:

- Points that are not on the path have a "Zero-Max" fitness, i.e. their fitness is their number of 0s;
- The first point on the path is $0, 0, \ldots, 0$ and has fitness ℓ;
- Any point on the path has exactly 2 neighbors on the path with Hamming distance 1; two consecutive points on the path have a fitness difference of 1;
- The length of the path is $(k+1)2^{(l-1)/k} - k + 1$;
- A mutation of $i < k$ bits can only lead to a point which is either off the path (hence with a very low fitness), or on the path but only i positions away from the original point; *shortcuts* (i.e. jumps to very distant points on the path) can only be achieved by mutating at least k bits; using a mutation operator which mutates every bit independently with probability p, the probability of finding a given shortcut is hence $p^k(1-p)^{l-k}$.

Long k-path problems are defined by recurrence on ℓ. Starting from the Long k-path $P(k, \ell)$, the $P(k, \ell + k)$ path is made of three parts: (i) the first part S_0 is made by concatenating k 0's to each point of $P(k, \ell)$; the third part S_1 is made by concatenating k 1's to each point of $P(k, \ell)$ in reverse order; S_0 and S_1 are linked by a "bridge" containing $(k-1)$ points, created by concatenating $0 \ldots 01, 0 \ldots 011, \ldots, 001 \ldots 1, 01 \ldots 1$ to the final point of $P(k, \ell)$. The original Long Path problem is a Long k-path with $k = 2$. The path length decreases as k increases, together with the probability of finding a shortcut.

After [27], shortcuts provably speed up the convergence if $k \leq \sqrt{\ell}$ (for higher values of k one should simply follow the path). In such cases, "exceptional properties of operators sometimes reflect EA behavior more accurately than average typical properties".

AOS and Long 3-Path Problems

The reported experiments consider $k = 3$ with ℓ ranging in $\{43, 49, 55, 61\}$. An additional mutation operator, the $3/\ell$ bit-flip (flipping each bit with probability $3/\ell$), has been added to the operator set.

Note that Long k-paths are challenging problems for AOS: when the parent individual belongs to the path, the 1-bit mutation improves the fitness by 1, with probability $1/\ell$ while all other mutation operators will fail to improve the fitness (reward 0) in the vast majority of cases. Experimentally, the *Adaptive Pursuit* AOS does not cope well with Long k-paths and will be omitted in the following; the best results are obtained for $p_{min} = .2$, i.e. for a uniform selection of the operators.

Results

By construction, some Long k-path runs can be "lucky" and discover the short-cuts, thus yielding large standard deviations in the performance. For instance, the optimal result obtained for $\ell = 49$ reaches the solution in 3590 ± 3327 generations (averaged over 50 runs). For this reason, the results will be described in terms of min and median number of generations needed to reach the solution (as opposed to, average and standard deviation).

Table 2 displays the results obtained for *Ex-DMAB*, *Avg-DMAB* and the baseline approaches: the Oracle one always selects the optimal operator (determined in the same way as for the One-Max problem) and the Naive one uniformly selects an operator in the operator set. The AOS setting is the best one found by the F-Race over all considered Long k-path.

Table 3 reports the results obtained for the best AOS setting, found by the F-Race over each considered Long k-path. The optimal setting $(W(C, \gamma))$ is indicated below the min and median number of generations to the solution. As could have been expected, the *Avg-DMAB* AOS goes for a medium window size $(W = 50)$ whereas the *Ex-DMAB* needs a much larger window size $(W = 500)$. Besides, the F-Race retains many good settings for the AOS parameters, suggesting that the C and γ parameters together control the Exploration vs Exploitation tradeoff, and might be redundant to some extent.

Table 2. Extreme vs Average Reward and DMAB AOS on the Long k-path, $k = 3$, min - median number of generations to the solutions out of 50 runs; the robust optimal AOS parameters $(W, (C, \gamma))$ are indicated below

ℓ	DMAB - $W(C, \gamma)$		Optimal	Uniform
	Extreme 500 (100; 100)	Average 50 (50; .5)		
43	11 - 2579	61 - 2342	2 - 1202	50 - 3393
49	17 - 4467	6 - 6397	19 - 2668	5 - 4904
55	161 - 6190	54 - 8222	45 - 3224	344 - 10068
61	251 - 13815	94 - 15304	8 - 5408	12 - 9590

Table 3. Extreme vs Average Reward and DMAB AOS on the Long k-path, $k = 3$, min - median number of generations to the solutions out of 50 runs, using the optimal AOS parameter for each ℓ

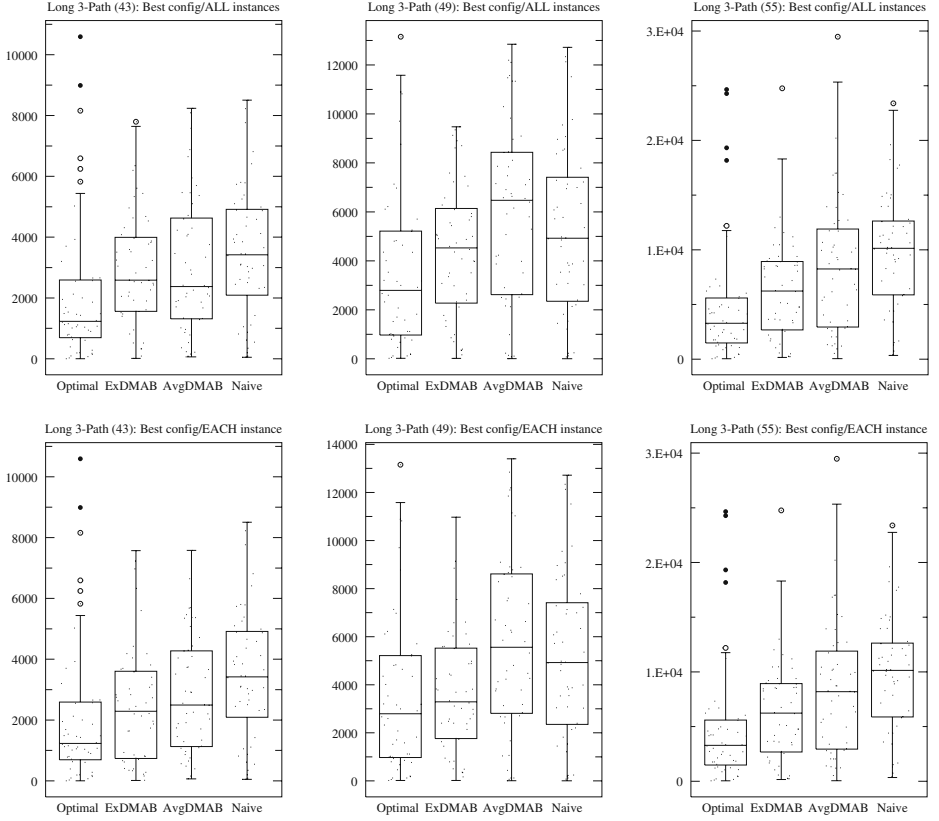| ℓ	DMAB - $W(C, \gamma)$		Optimal	Uniform
	Extreme	Average		
43	11 - 2216 500(50; 50)	66 - 2487 50(.5; 100)	2 - 1202	50 - 3393
49	17 - 3244 500(100; 500)	6 - 5321 50(.1; 1000)	19 - 2668	5 - 4904
55	161 - 6190 500(100; 100)	54 - 8158 50(50; .1)	45 - 3224	344 - 10068
61	80 - 10253 500(50; 25)	94 - 13865 50(.5; 50)	8 - 5408	12 - 9590

Fig. 3. AOS results on Long k-paths, $k = 3$

The significance of these results is assessed using unsigned Wilcoxon rank sum and Kolmogorov-Smirnov non-parametric tests (thereafter referred to as W and KS).

In the robust scenario (AOS parameters are selected by F-race over all Long k-path problems), *Ex-DMAB* is outperformed by (respectively similar to) the Oracle AOS for $\ell \in \{43, 61\}$ (resp. $\ell \in \{49, 55\}$) with confidence 99% according to both W and KS tests. *Ex-DMAB* outperforms *Avg-DMAB* for $\ell = 49$ (with W at 90% and K at 95%) and for $\ell = 55$ (with W at 90%).

In the fine tuning scenario (AOS parameters are selected by F-race for each Long k-path problem), *Ex-DMAB* obtains better results as could have been expected: no significant difference between *Ex-DMAB* and the Oracle strategy is observed, with confidence 99% according to both W and KS tests. In the meanwhile, *Ex-DMAB* significantly improves on the Naive AOS in all instances ($\ell = 43$, for W at 99% and KS at 95%; $\ell = 49$, for W at 95% and KS at 90%; $\ell = 55$, for W and KS at 99%), except for the $\ell = 61$ one. Comparatively to

Avg-DMAB, *Ex-DMAB* obtains similar (respectively significantly better) performances for $\ell \in \{43, 61\}$ with both tests at 99% (resp. for $\ell = 49$, with W at 99% and KS at 95%; for $\ell = 55$, with W at 90%).

The empirical distributions of all approaches are displayed on Fig. 3. The case $\ell = 61$ is omitted as no strategy was found effective on this problem, which is blamed on the very low probability of finding shortcuts.

5   Discussion and Perspectives

This paper provides a proof of principle for the proposed *Ex-DMAB* Adaptive Operator Selection, based on ample empirical evidence gathered from the One-Max and Long k-path problems. *Ex-DMAB* was found to efficiently detect the best mutation operators during the whole course of evolution, keeping up with the Oracle strategy in the majority of cases.

Although its good performances rely on the expensive offline tuning of *Ex-DMAB* parameters, *Ex-DMAB* was found to outperform the main options opened to the naive EA user, namely (i) using a fixed or deterministic strategy (including the naive, uniform selection, strategy; (ii) using a former AOS strategy. Furthermore, *Ex-DMAB* involves a fixed and limited number of parameters (the window size W, the scaling factor C and the change detection test threshold γ), whereas the number of operator rates increases with the number of operators.

The most challenging situations for *Ex-DMAB* are the last stages of evolution. At this point, the best operator hardly brings any improvement, and *Ex-DMAB* is found to tend toward the uniform naive strategy (uniformly selecting an operator in the pool). A tentative interpretation for this fact is as follows. On the one hand, fitness improvements are more and more rare with respect to the window size, leading to uniform rewards and hence to uniform selection. Furthermore, when a reward occurs after a long wandering period, it is likely to trigger the change detection test, causing the Dynamic Multi-Armed Bandit to restart from scratch, thus increasing the exploration bias.

Further research will aim at addressing the above weaknesses. A first perspective is opened by learning across runs, specifically recording the statistics of fitness improvement in relation with the current average fitness; these statistics will serve to adjust the window length W in the last stages of evolution. A second perspective is to adjust online the Page Hinkley parameter γ, depending on the estimated number of transitions (change of the best operator) in the fitness landscape. Along the same lines, we shall investigate how γ and the scaling factor C relate, as both cooperate to control the exploration vs exploitation trade-off.

References

1. Grefenstette, J.: Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics 16(1), 122–128 (1986)
2. Lobo, F., Lima, C., Michalewicz, Z. (eds.): Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence, vol. 54. Springer, Heidelberg (2007)
3. Davis, L.: Adapting operator probabilities in genetic algorithms. In: Schaffer, J.D. (ed.) Proc. ICGA 1989, pp. 61–69. Morgan Kaufmann, San Francisco (1989)
4. Da Costa, L., Fialho, A., Schoenauer, M., Sebag, M.: Adaptive operator selection with dynamic multi-armed bandits. In: Keijzer, M. (ed.) Proc. GECCO 2008, pp. 913–920. ACM Press, New York (2008)
5. Fialho, A., Da Costa, L., Schoenauer, M., Sebag, M.: Extreme value based adaptive operator selection. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 175–184. Springer, Heidelberg (2008)
6. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning 47(2-3), 235–256 (2002)
7. Hinkley, D.: Inference about the change point from cumulative sum-tests. Biometrika 58(3), 509–523 (1971)
8. Rudolph, G.: Convergence Properties of Evolutionary Algorithms. Verlag Dr. Kovac (1997)
9. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation 3(2), 124–141 (1999)
10. Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E.: Parameter control in Evolutionary Algorithms. In: Lobo, F.G., et al. (eds.) Parameter Setting in Evolutionary Algorithms, pp. 19–46. Springer, Heidelberg (2007)
11. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W.B., et al. (eds.) Proc. GECCO 2002, pp. 11–18. Morgan Kaufmann, San Francisco (2002)
12. Yuan, B., Gallagher, M.: Statistical racing techniques for improved empirical evaluation of evolutionary algorithms. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 172–181. Springer, Heidelberg (2004)
13. Bartz-Beielstein, T., Lasarczyk, C., Preuss, M.: Sequential parameter optimization. In: McKay, B. (ed.) Proc. CEC 2005, pp. 773–780. IEEE Press, Los Alamitos (2005)
14. Nannen, V., Eiben, A.E.: Relevance estimation and value calibration of evolutionary algorithm parameters. In: Veloso, M. (ed.) Proc. IJCAI 2007, pp. 975–980 (2007)
15. De Jong, K.: Parameter Setting in EAs: a 30 Year Perspective. In: Lobo, F.G., et al. (eds.) Parameter Setting in Evolutionary Algorithms, pp. 1–18. Springer, Heidelberg (2007)
16. Lobo, F., Goldberg, D.: Decision making in a hybrid genetic algorithm. In: Porto, B. (ed.) Proc. ICEC 1997, pp. 121–125. IEEE Press, Los Alamitos (1997)
17. Tuson, A., Ross, P.: Adapting operator settings in genetic algorithms. Evolutionary Computation 6(2), 161–184 (1998)
18. Barbosa, H.J.C., Sá, A.M.: On adaptive operator probabilities in real coded genetic algorithms. In: Workshop on Advances and Trends in AI for Problem Solving – SCCC 2000 (2000)

19. Julstrom, B.A.: What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm on genetic algorithms. In: Eshelman, L.J. (ed.) Proc. ICGA 1995, pp. 81–87. Morgan Kaufmann, San Francisco (1995)
20. Maturana, J., Saubion, F.: A compass to guide genetic algorithms. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 256–265. Springer, Heidelberg (2008)
21. Whitacre, J.M., Pham, T.Q., Sarker, R.A.: Use of statistical outlier detection method in adaptive evolutionary algorithms. In: Keijzer, M. (ed.) Proc. GECCO 2006, pp. 1345–1352. ACM Press, New York (2006)
22. Goldberg, D.E.: Probability matching, the magnitude of reinforcement, and classifier system bidding. Machine Learning 5(4), 407–425 (1990)
23. Thierens, D.: An adaptive pursuit strategy for allocating operator probabilities. In: Beyer, H.G. (ed.) Proc. GECCO 2005, pp. 1539–1546. ACM Press, New York (2005)
24. Wong, Y.Y., Lee, K.H., Leung, K.S., Ho, C.W.: A novel approach in parameter adaptation and diversity maintenance for genetic algorithms. Soft Computing 7(8), 506–515 (2003)
25. Lai, T., Robbins, H.: Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics 6(1), 4–22 (1985)
26. Horn, J., Goldberg, D.E., Deb, K.: Long path problems. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 149–158. Springer, Heidelberg (1994)
27. Garnier, J., Kallel, L.: Statistical distribution of the convergence time of evolutionary algorithms for long-path problems. IEEE Transactions on Evolutionary Computation 4(1), 16–30 (2000)