Adaptive Operator Selection with Dynamic Multi-Armed Bandits

Luis DaCosta[1], Álvaro Fialho[2], Marc Schoenauer[1,2], Michèle Sebag[1,2]
[1]Team TAO, LRI (UMR CNRS 8623)
INRIA Saclay - Île-de-France
Bat 490, Université Paris-Sud
91405 Orsay Cedex, France
[2]Microsoft Research–INRIA Joint Centre
Parc Orsay Université
28, rue Jean Rostand
91893 Orsay Cedex, France
FirstName.LastName@inria.fr

ABSTRACT

An important step toward self-tuning Evolutionary Algorithms is to design efficient Adaptive Operator Selection procedures. Such a procedure is made of two main components: a credit assignment mechanism, that computes a reward for each operator at hand based on some characteristics of the past offspring; and an adaptation rule, that modifies the selection mechanism based on the rewards of the different operators. This paper is concerned with the latter, and proposes a new approach for it based on the well-known Multi-Armed Bandit paradigm. However, because the basic Multi-Armed Bandit methods have been developed for static frameworks, a specific Dynamic Multi-Armed Bandit algorithm is proposed, that hybridizes an optimal Multi-Armed Bandit algorithm with the statistical Page-Hinkley test, which enforces the efficient detection of changes in time series. This original Operator Selection procedure is then compared to the state-of-the-art rules known as *Probability Matching* and *Adaptive Pursuit* on several artificial scenarios, after a careful sensitivity analysis of all methods. The Dynamic Multi-Armed Bandit method is found to outperform the other methods on a scenario from the literature, while on another scenario, the basic Multi-Armed Bandit performs best.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Adaptivity, Operator selection, Multi-Armed Bandit

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

1. PARAMETER CONTROL IN EAS

Parameter control has become a hot research area in many domains of Computer Science, and in particular in Optimization. While optimization methods are widely used in different application areas, it remains that an extensive expertise and/or extensive experimentations are needed to fully benefit from their possibilities. Like all high technology techniques, optimization methods thus require specific developments in order to *cross the chasm* [16] and make it outside of research labs. Concretely, these methods should become self-tuning, i.e. able to automatically parametrize themselves, in order to be accessible to a wide audience without any knowledge in the optimization technique itself.

The above remarks indeed apply to Evolutionary Algorithms (EAs); although their many parameters are appreciated for the great search flexibility they enforce, setting the parameter values best suited to the problem at hand defines an optimization problem *per se*, as noted in the very early days of Evolutionary Computation [9].

Accordingly, the topic of parameter-tuning has been extensively investigated in the literature, and is still an active field of research [15]. Amongst the different approaches to parameter setting, this paper is concerned with Parameter Control (following the terminology of [6, 7]), which proceeds by modifying the parameters along the EA run. Along the same lines, three types of Parameter Control are distinguished: in *deterministic control*, parameter values are predefined functions of time; in *self-adaptive control*, parameters are attached to the individual genotype and their values are set by evolution; in *adaptive control*, parameter values are predefined functions of the whole history of the run. While deterministic control suffers from the same difficulties as static control (finding the optimal pre-defined functions would require Parameter Tuning), self-adaptive and adaptive control have met some success along the years, but only in very specific domains, like in the framework of continuous parameter optimization (see the discussion in [5] and references herein).

One key objective of Parameter Control is thus to propose generic methods for the control of representation independent parameters: i) population size, ii) selection mode and parameters, iii) variation operator probabilities.

This paper is focused on the latter issue, or more generally, on *Adaptive Operator Selection* (AOS): modifying on-line the procedure that chooses a given variation operator among the set of operators relevant to the problem representation, an issue faced by all EAs. Section 2 will briefly review related

work, as many approaches have been proposed since Davis' seminal work [4].

In essence, the goal of AOS is to select the operator which maximizes the (expectation of) fitness improvement[1]. AOS thus raises two main difficulties. The first one might be seen as an Exploration vs Exploitation (EvE) dilemma; some exploration is needed to find the truly best operator, while the best operators found so far should be selected as often as possible in order to maximize the *cumulative* improvement. The EvE dilemma has been intensively studied in the context of Game Theory, in the so-called Multi-Armed Bandit (MAB) framework [13, 1] (see Section 3). The second difficulty is that the best operator does not need to be the same at different stages of evolution, and the quality of a specific variation operator (however measured) usually varies along the evolutionary search. Formally, the problem is thus a Dynamic Multi-Armed Bandit (D-MAB) setting.

In this context, the contribution of the paper is a control procedure addressing the D-MAB settings. Specifically, it hybridizes an optimal MAB algorithm first proposed by [1], with the statistical Page-Hinkley test [10], enforcing the efficient detection of changes in time series. This methodology is experimentally validated and compared, in terms of both performance and robustness, with the state-of-art *Probability Matching* [8, 14, 2] and *Adaptive Pursuit* [20], following the protocol proposed in [20], i.e. using artificial scenarios, independently of any Evolutionary Algorithm.

The paper is organized as follows. Section 2 presents a survey of AOS procedures. Section 3 details several decision making algorithms: First, the (static) MAB framework and the corresponding state-of-art algorithms; then, the state-of-the-art dynamic decision making techniques that have been used within AOS methods, namely *Probability Matching* and *Adaptive Pursuit*. D-MAB is then introduced and discussed in section 4. Section 5 describes the experimental validation setting; the artificial reward scenario used to validate the *Probability Matching* and *Adaptive Pursuit* procedures in [20, 21] is extended, and some variants are proposed to investigate the robustness of the control. Section 6 reports on the comparative validation of D-MAB with respect to *Probability Matching*, *Adaptive Pursuit* and the static MAB algorithm; a detailed sensitivity analysis of all four approaches is presented. The findings of this work and further directions of research are discussed in section 7.

2. ADAPTIVE OPERATOR SELECTION

Adaptive methods (sometimes also called *feedback methods*), use information from the history of evolution to modify some parameters. There are two main components for an AOS method: the credit assignment (how to assess the goodness of an operator?), and the decision making (based on past credits, how to choose the next operator to use?). This section will survey existing adaptive methods for operator selection, looking at how they addressed these issues.

2.1 State-of-the-art

The first AOS procedure was proposed as early as in the late 80s by D. Davis [4]. In this work, the probability of each operator is adjusted based on how often its application helped improving the best fitness in the population, with a propagation mechanism toward the parents of the newborn best individual: an operator is considered beneficial if it created individuals that in turn were the parents of improved offspring, and so forth, with a decaying mechanism. Similar ideas were applied by Julstrom [12] with a few significant differences: an operator was considered beneficial if the offspring it created was better than its own parents, and not only than the current best of the population. Moreover, the propagation of reward to the parents of the good offspring was simpler than that proposed by Davis. Some good results are reported in both works, compared to static parameters. And in both cases, beside the complex reward mechanism, the update mechanism for the operator probabilities is somehow similar to the Probability Matching method (see below), in that the probability of an operator is proportional to an estimate of its reward with decay mechanism.

A different approach was used in the *CORBA* (Cost Operator Based Rate Adaptation) method [22], where operators probabilities are periodically readjusted based on their "productivity" in the previous generations. Such productivity is defined by the average improvement of the offspring over their parents during the time period (for those offspring that were better than their parents) divided by the computational cost of evaluating a child. One main difference in relation to both works cited above is that no credit is given upward in the generations. Moreover, the user is asked to define a fixed list of possible values for operator probabilities, and every G generations, the rank of the operator productivity determines the value in the list that will be assigned to its probability. Beside a favorable comparison with some self-adaptive method, the authors report that, though the adaptation is beneficial on some problems, it might not be the case for all problems. Moreover, an important factor is the cost of the adaptation itself, that should be carefully balanced with the improvement it brings to the search – if any.

In [2], the authors use a Probability Matching technique to adapt the operator probabilities in the context of a real-coded GA. Their main contribution (apart from trying out the idea in a continuous context) is to use a mixture of local reward (i.e. estimated from the improvement of fitness of the offspring over its parents) and global measure of performance (i.e. obtained by comparing the offspring fitness to some statistics about the fitness in the whole population). As in [22], the cost (number of function evaluations required, if any) is also used to decrease the reward of the operator. The adaptation rule is very close to the Probability Matching method.

2.2 Discussion

A critical issue in the above works is the definition of the credit deserved by an operator. More precisely, *what characterizes a good operator?* A second issue regards the credit assignment procedure, aimed to find the best strategy of operator probability control.

Given the difficulty of the task, it might seem relevant to separately investigate both issues, namely the definition of the operator reward, and the search for an efficient AOS strategy. A first step along this line is taken by [20], tackling the control of operator probabilities in artificial dynamic environments, assuming the reward associated to each op-

[1]More precisely, the goal would to be to select the right mixture of operators [19]. This mixture selection problem can in principle be handled in the AOS setting, by considering mixtures as particular options; this point will not be considered in the scope of this paper.

erator to be known. This framework is quite similar to the so-called Multi-Armed Bandit framework; both frameworks are presented in the next section.

3. DECISION MAKING

This section first introduces the static Multi-Armed Bandit (MAB) setting, referring the reader to [1] for a more comprehensive presentation. The closely related dynamic setting considered by [20] is described thereafter, together with the *Probability Matching* and *Adaptive Pursuit* methods.

3.1 Static Setting

A multi-armed bandit involves K arms. The i-th arm is characterized by its (fixed, unknown) reward probability p_i ($p_i \in [0,1]$). At each time step t, the player (game strategy) selects some arm j; with probability p_j it gets reward $r_t = 1$, otherwise $r_t = 0$.

At any point T during the game, the performance of the strategy is measured after its cumulative reward $\sum_{t=1}^{T} r_t$. An equivalent criterion, more amenable to theoretical analysis, is the so-called *regret* of the strategy, defined as the difference between its performance and the best possible performance; clearly, the best possible performance is achieved by playing at each time step the (unknown) arm with maximal reward p^*. Hence the regret of the strategy after T time steps is:

$$\mathcal{L}(T) = Tp^* - \sum_{t=1}^{T} r_t$$

Classically, it is assumed that i) arms are independent from each other; ii) the rewards associated to each arm are independently and identically distributed (iid). Under these assumptions, it is shown that the optimal regret logarithmically increases with time ($\mathcal{L}(T) = \mathcal{O}(\log(T))$) [13].

The so-called *Upper Confidence Bound* (UCB) algorithm devised by Auer et al. [1] achieves the optimal regret rate above through an Exploration vs Exploitation-based criterion. Formally, let $n_{i,t}$ denote the number of times the i-th arm has been played up to time t, and let $\hat{p}_{i,t}$ denote the average corresponding reward; subscript t will be omitted when clear from the context. The UCB1 algorithm selects in each time step t the arm maximizing the quantity below:

$$\hat{p}_{j,t} + \sqrt{\frac{2\log\sum_k n_{k,t}}{n_{j,t}}} \quad (1)$$

Clearly, the first term favors Exploitation (play the arm with best empirical reward) while the second term enforces Exploration, each arm being selected infinitely many times as t goes to infinity. However the laps of time between two selections of some under-optimal arm increases exponentially.

Some adjustments of the Exploration term have been proposed in the literature and are summarized in Table 1. UCB-Tuned (UCBT) [1] takes into account the variance of the rewards, often experimentally outperforming UCB1, though with no formal guarantee. KUCBT [11] only differs from UCBT as term $\log(\sum_k n_{k,t})/n_{j,t}$ has been omitted. Lastly, *cUCB* allows one to directly control the exploration strength through constant c. Typically, when the number of arms is large and/or when the time horizon is restricted it makes sense to limit the exploration strength.

$\hat{p}_j + \sqrt{\frac{2log\sum_k n_{k,t}}{n_{j,t}}}$	UCB1 [1]
$\hat{p}_j + \frac{log\sum_k n_{k,t}}{n_{j,t}} + \sqrt{\frac{v_{j,t}log\sum_k n_{k,t}}{n_{j,t}}}$	UCB-Tuned [1]
$\hat{p}_j + \sqrt{\frac{v_{j,t}log\sum_k n_{k,t}}{n_{j,t}}}$	KUCBT [11]
$\hat{p}_j + \sqrt{\frac{clog\sum_k n_{k,t}}{n_{j,t}}}$	*cUCB*

Table 1: UCB Variants, where $v_{j,t} = max(\varepsilon, V_{j,t})$ and $V_{j,t}$ is the empirical variance of the reward for the j-th arm. $v_{j,t}$ is used instead of $V_{j,t}$ (ε being a small constant) to avoid definitively rejecting arm j in case of a null variance.

It must be noted that Multi-Armed Bandits differ in two respects from the mainstream framework concerned with learning optimal strategies, namely Reinforcement Learning (RL). On the one hand, MAB aims to select the best action, whereas RL is concerned with finding the best sequence of actions. On the other hand, while RL is concerned with learning the optimal sequence, it does not pay attention to the rewards it gets during the training phase. Quite the contrary, MAB simultaneously wants to learn the best action, and to optimize the cumulative reward it gets *during learning*. Clearly, the latter objective is the only one relevant in the context of Parameter Control: the goal is to learn the best operators along evolution while maximizing the fitness improvement in the course of evolution.

3.2 Dynamic Setting

Interestingly, the setting proposed by Thierens [20, 21] in order to investigate the various issues raised by AOS in a dynamic environment resembles that of the Multi-Armed Bandit, except that it is dynamic. Formally, a reward distribution, uniform in a given interval $[p_{i,t} - 1, p_{i,t} + 1]$, is associated to the i-th arm or operator. Further, the center $p_{i,t}$ of this reward distributions varies every P time steps (in the experiments $P = 50$ or $P = 200$); the reward distributions associated to all operators change simultaneously.

Indeed, it is doubtful that the reward of variation operators (however measured) obeys such simple distributions. Nevertheless, the construction of efficient AOS strategies in this simplified setting undoubtedly is a significant milestone toward adaptive self-tuning EAs.

Along these lines, [20] aims to identify at every time step the optimal selection probability $s_{i,t}$ for every operator i ($i = 1$ to K, with $\sum_{i=1}^{K} s_{i,t} = 1$), such that it maximizes the expected cumulative reward at the end of the run. Empirically, the cumulative reward is obtained by selecting in each time step t the i-th operator with probability $s_{i,t}$, and recording the associated reward.

Like in MAB framework, [20] uses and maintains an estimate of the current operator reward noted $\hat{p}_{i,t}$. At time t:
• arm i is selected with probability $s_{i,t}$
• the corresponding reward r_t is drawn uniformly in $[p_{i,t} - 1, p_{i,t} + 1]$;
• the reward estimate $\hat{p}_{i,t}$ of the selected arm is updated after r_t, using an additive relaxation mechanism with learning rate α ($0 < \alpha \leq 1$). Parameter α thus controls the memory of the reward estimate; the forgettingness increases with α.

$$\hat{p}_{i,t+1} = (1-\alpha)\hat{p}_{i,t} + \alpha\, r_t \quad (2)$$

Based on the reward estimates, two methods named *Prob-*

ability Matching and *Adaptive Pursuit* have been proposed to adjust the selection probabilities $s_{i,t}$.

The *Probability Matching* by far the most popular amongst AOS methods [8, 14, 2] (see also Section 2) preserves the Exploration vs Exploitation balance by i) defining a minimal selection probability p_{min} ($\forall t, \forall i, s_{i,t} \geq p_{min}$) and ii) making $s_{i,t} - p_{min}$ proportional to $\hat{p}_{i,t}$. *Probability Matching* thus involves two parameters, the learning rate α used to update the reward estimate $\hat{p}$, and the minimal selection probability p_{min}.

$$s_{i,t+1} = p_{min} + (1 - K * p_{min}) \frac{\hat{p}_{i,t+1}}{\sum_{j=1}^{K} \hat{p}_{j,t+1}} \tag{3}$$

The *rationale* behind *Probability Matching* is that the selection probability of any operator should not become too low, preventing *de facto* AOS from discovering that this operator has become the optimal one. Except for this reservation, the selection probability of every operator is meant to reflect the associated reward.

After equations (2) and (3), if the i-th operator does not get any reward for some period of time, $\hat{p}_{i,t}$ goes to 0 while $s_{i,t}$ goes to p_{min}. Inversely, if the i-th operator gets all rewards, $s_{i,t}$ goes to $p_{max} = 1 - K * p_{min}$. In the meanwhile, all mildly relevant operators ($0 < s_{i,t} < max\{s_{i,t}\}$) keep being selected, thus hindering the performance of *Probability Matching* [20].

This drawback is addressed by the *Adaptive Pursuit* method. Originally proposed for learning automata, this method aims to select at each time step the operator i^*_t with maximal estimated reward ($i^*_t = argmax\{\hat{p}_{i,t}\}$). To do so, the update mechanism follows a "winner take all" strategy, increasing the selection probability of the best arm up to p_{max} while all other selection probabilities are decreased to p_{min}.

$$\left\{ \begin{array}{ll} i^* & = argmax\{\hat{p}_{i,t}, i = 1 \ldots K\} \\ s_{i^*,t+1} & = s_{i^*,t} + \beta \left(p_{max} - s_{i^*,t}\right), \\ s_{i,t+1} & = s_{i,t} + \beta \left(p_{min} - s_{i,t}\right), \text{ for } i \neq i^* \end{array} \right. \tag{4}$$

Adaptive Pursuit thus involves one additional parameter compared to *Probability Matching*, the learning rate β used to update the selection probabilities.

4. DYNAMIC MULTI-ARMED BANDIT

As shown in section 3.1, Multi-Armed Bandits are not suited to dynamic environments. Typically, if the current best arm becomes less efficient and dominated by another one, it might take a long time before the latter arm catches up. The proposed D-MAB algorithm thus combines MAB ideas with a specific statistical test known as Page-Hinkley (PH), which is used to detect the changes in the environment and, accordingly, restart the multi-armed bandit.

After describing the PH test, this section gives an overview of D-MAB. The UCB algorithm presented in section 3.1 is adapted to handle continuous rewards as in [20], whereas the standard MAB framework considers boolean rewards.

4.1 Change-point Detection

Given a series of observations $r_1, \ldots, r_\ell$, a frequently asked question is if these observations can be attributed to a same statistical law (null hypothesis) or if some change in the law underlying the observations has occurred at some point (*Change* hypothesis).

A standard test for the change hypothesis is the Page-Hinkley (PH) statistics [17]. Let $\bar{r}_\ell$ denote the average of $r_1, \ldots r_\ell$ and let e_ℓ denote the difference $r_\ell - \bar{r}_\ell + \delta$, where δ is a tolerance parameter [18]. The PH test considers the random variable m_t defined as the sum of $e_1, \ldots, e_t$. The maximum value M_t of the m_t for $\ell = 1 \ldots t$ is also computed and the difference between M_t and m_t is monitored; when this difference is greater than some user-specified threshold λ, the PH test is triggered, i.e., it is considered that the *Change* hypothesis holds.

$$\begin{array}{l} \bar{r}_\ell = \frac{1}{\ell} \sum_{i=1}^{\ell} r_i \\ m_t = \sum_{\ell=1}^{t} (r_\ell - \bar{r}_\ell + \delta) \\ M_t = \max\{m_\ell, \ \ell = 1 \ldots t\} \\ PH_t = M_t - m_t \\ \text{Return } (PH_t > \lambda) \end{array} \tag{5}$$

The PH test involves two parameters. Parameter λ controls the trade-off between false alarms and un-noticed changes. Parameter δ enforces the robustness of the test when dealing with slowly varying environments.

4.2 D-MAB

D-MAB hybridizes the UCB1 algorithm (Section 3.1) and the Page-Hinkley test by restarting UCB1 from scratch when the PH test is triggered. Formally, D-MAB maintains four indicators for each arm i: the number n_i of times this arm has been played up to time t; the average empirical reward $\hat{p}_i$; the average and maximum deviation m_i and M_i involved in the PH test, initialized to 0 and updated as detailed below. At each time step t:

- D-MAB selects an arm i after equation (1). It receives some reward r_t, drawn after reward distribution $p_{i,t}$.
- All four indicators $\hat{p}_i, n_i, m_i$ and M_i are updated accordingly:

$$\begin{array}{rl} \hat{p}_i := & \frac{1}{n_i+1}(n_i \hat{p}_i + r_t) \\ n_i := & n_i + 1 \\ m_i := & m_i + (\hat{p}_i - r_t + \delta) \\ M_i := & max(M_i, m_i) \end{array}$$

- If the PH test is triggered ($M_i - m_i > \lambda$), the bandit is restarted, i.e., for all arms, all indicators $n_i, \hat{p}_i, m_i$ and M_i are set to 0.

Instead of reinitializing $\hat{p}_i$ and n_i to 0 when the PH test is triggered, one might consider to discount their values, e.g. using some relaxation mechanism as in *Probability Matching*. Empirically, it turns out however that the restart mechanism is more robust. A tentative explanation for this fact goes as follows. While UCB, as it is, can handle slowly varying reward distributions, its real limitation is due to its inertia in the case where the current best arm is well established and undergoes a sudden fall[2]. In such cases, the simple restart mechanism enables to recover accurate estimates of the rewards much faster than relaxation.

[2]When the current best arm has been selected n_i times and its reward falls down by δ, it needs $n_i \delta / \varepsilon$ moves before recovering an accurate reward estimate up to precision ε.

4.3 Reward Scaling

A second difference between the MAB setting and the one considered by [20], besides the dynamic aspects, is that MAB considers boolean rewards and Bernouilli reward distributions while [20] considers continuous rewards and uniform reward distributions. The order of magnitude of the empirical reward (the Exploitation-related term in Eq. (1)) might thus be quite different from that of the Exploration term, adversely affecting the Exploration/Exploitation balance in UCB1. Accordingly, two scaling mechanisms proposed to control the EvE balance are incorporated in D-MAB.

Multiplicative Scaling (cUCB). The simplest scaling option is to multiply all rewards by a fixed user-defined parameter $C_{M-scale}$. This option relies on the user's prior knowledge about the range of the rewards.

Affine Scaling. The other scaling option is inspired by the well-known Fitness Scaling heuristics, used to calibrate the roulette wheel selection in Canonical GA. The Affine Scaling option similarly relies on the user's prior knowledge about the desired selection pressure, here the selection probability of the best operator. Formally, Affine Scaling computes at every time-step two coefficients a and b such that the scaled empirical rewards $\hat{q}_{i,t} = a\hat{p}_{i,t} + b$ satisfy[3]:

$$\begin{cases} \sum_{i=1}^{K} \hat{q}_{i,t} = 1 \\ \max_i \hat{q}_{i,t} = C_{A-scale} \end{cases}$$

where $C_{A-scale}$ is a positive, user-defined constant. Empirically, $C_{A-scale}$ should take values around 1; it can even exceed 1, leading to negative scaled empirical rewards for the worst arms. Clearly, the selection rule (Eq. 1) only depends on coefficient a; the difference compared to Multiplicative Scaling is that a is adjusted in every time step depending on the relative ranges of the operator rewards.

Finally, D-MAB involves four parameters: parameters λ and δ used in the PH test; the type of scaling (multiplicative or affine); and the scaling factor $C_{M-scale}$ or $C_{A-scale}$ depending on the type of scaling.

5. EXPERIMENTAL SETTING

This section describes the goal of the experiments, and the methodology followed to experimentally assess the various AOS algorithms.

5.1 Goal of the Experiments

The goal of the experiments is to assess the performance of all AOS algorithms below:

- The *Probability Matching* and *Adaptive Pursuit* methods (section 3.2, [21]) will be used as baseline algorithms. *Probability Matching* parameters include the minimal selection probability p_{min} and the relaxation factor α; *Adaptive Pursuit* involves an additional parameter, the learning rate β.

- The UCB algorithm [1] proposed for static Multi-Armed Bandit problems (section 3.1) will also be considered as baseline, combined with the multiplicative and affine scaling mechanisms (section 4.3). For the sake of clarity, the multiplicative (respectively, affine) variants will be referred to as MAB-M (resp., MAB-A). Both MAB-M and MAB-A involve a single parameter, the scaling constant $\mathcal{S}$.

- Finally, the D-MAB algorithm described in section 4 will also be considered, combined with the multiplicative and affine scaling mechanisms, referred to as D-MAB-M and D-MAB-A. Both algorithms involve three parameters: the λ and δ parameters used in the PH test (section 4.1) and the scaling constant $\mathcal{S}$.

Two performance measures will be considered. The first one, noted $\hat{p}_{\text{best}}$, is the probability of picking up the best operator, estimated as the percentage of time steps the operator with current best reward is selected [20], averaged over 100 independent runs for each parameter setting. The second one is the total cumulative reward (TCR), estimated as the sum of the rewards gathered along the run and likewise averaged over 100 independent runs. It is claimed that TCR better reflects the priorities of evolution than $\hat{p}_{\text{best}}$: maximizing the cumulative reward would correspond to optimizing the best fitness in the run if the operator rewards directly reflected the fitness improvement (see below).

5.2 Experiment scenarios

The first scenario considered is the one defined in [20]. Two other variants will be investigated in order to assess the robustness of the algorithms with respect to other reward distributions.

Formally, the number K of operators considered is set to 5. The reward of each operator follows a known probability distribution, which is constant along a fixed number of time steps ΔT, or epoch. At the end of each epoch (i.e. every ΔT time steps), all rewards are exchanged amongst operators following a given permutation. All runs in all scenarios involve 10 epochs and use the same series of 10 permutations. Two cases are considered: short epochs ($\Delta T = 50$) and long epochs ($\Delta T = 200$).

In the **Uniform Scenario** [20], distributions of the rewards are uniform distributions on different intervals. The best operator receives a reward that is uniformly distributed on $[4, 6]$, the second best on $[3, 5]$, etc, and the worst operator on $[0, 2]$. Note that those intervals are overlapping; the second best operator occasionally gets better rewards than the best operator.

In the **Boolean Scenario**, the rewards are either 0 or 10. The best operator receives reward 10 with probability .5, the second best with probability .4 and so forth. While the average reward for a given arm during a given epoch is the same as for the uniform scenario, the variance is much higher in the boolean scenario. This scenario is closer to the standard MAB setting (up to a factor of 10).

In the **Outlier Scenario**, all operators receive reward 0 with probability .9; the best operator receives reward 50 with probability .1, the second best receives reward 40 with probability .1, and so forth. Likewise, the average rewards in this scenario are the same as for the previous ones, but with a much higher variance.

In all scenarios, the optimal strategy, selecting the best operator in each time step, would get an average reward of 5 per time step; the maximal TCR thus is 2500 in the case of short epochs ($10 \times \Delta T \times 5$, $\Delta T = 50$) and 10 000 for long epochs ($\Delta T = 200$),

[3]It is seen that a and b are well defined except for the case where all $\hat{p}_{i,t}$, or all but 1, are 0.

6. EXPERIMENTAL RESULTS

This section first reports on the sensitivity analysis of every considered algorithm, and identifies its best parameter values. The performances of all algorithms (each one at its best) are then compared and discussed.

6.1 Design of Experiments

For the sake of a fair comparison, the optimal parametrization of all six algorithms has been determined through a factorial design of experiments, varying the parameters as follows (100 independent runs are launched for each parameter setting):

- For the *Probability Matching* and *Adaptive Pursuit* methods, the minimal selection probability p_{min} is varied in $[0, .1]$ by .01 increment. The relaxation factor α ranges in $[.1, 1]$ with .1 increment. For *Adaptive Pursuit*, parameter β similarly ranges in $[.1, 1]$ with .1 increment.
- For MAB algorithms with Multiplicative (respectively, Affine) Scaling, the scaling factor $\mathcal{S}$ ranges in $[2, 10]$ (respectively, $[.1, 2]$) with .1 increment.
- For dynamic MAB, the PH threshold λ ranges in $[0.5, 10]$ with 0.5 increment in the Uniform scenario, and in $[10, 20]$ by 2 increment in the other scenarios. Parameter δ was set to .15 after a few preliminary experiments.

The analysis of the results is based on a 1-way ANOVA with confidence 95%, ordering the parameters of each algorithm by decreasing impact on the performances. If the influence of the first parameter, say p_1, is statistically significant, its best value p_1^* is determined using a Scheffé's test. A similar procedure is used to determine the best value for the second parameter conditioned by $p_1 = p_1^*$, and so forth (note that all algorithms have two parameters, except *Adaptive Pursuit* that has three).

The optimal parameter values (or range of values) and the associated performances are reported for all six algorithms in Tables 2 and 3, distinguishing the Uniform scenario (top), the Boolean scenario (middle) and the Outlier scenario (bottom).

The stability of the performances can be graphically assessed after the response surface of the algorithms with respect to their most influential parameters. Figures 1 and 2 respectively display the behavior of D-MAB-A and *Adaptive Pursuit*, where each point represents the average TCR over 100 runs.

6.2 Sensitivity Analysis

A first finding is that the *Outlier* scenario is too harsh for all considered algorithms; the best $\hat{p}_{\text{best}}$ is close to random guessing and the surface response (not shown due to space constraints) is chaotic. This scenario will thus be not discussed further in this paper.

In other scenarios, the MAB algorithms show good performances. Figure 1 is a typical plot of the response surface for an algorithm of the MAB family, that exhibits a rather stable area of high values. The landscape is even smoother for the *Uniform* scenario, and slightly more rough for the case $\Delta T = 50$, but with the same general outlook. Extreme values of λ (not visible on the plot) degrade the results; for medium values of λ, the results seem not very sensitive to the value of the scaling factor $\mathcal{S}$.

Regarding the *Probability Matching* and *Adaptive Pursuit* algorithms, a first surprising result is that the optimal p_{min} value is very low, .01 or 0 (Tables 2 and 3, Figure 2), whereas the original experiments reported in [20] used value .1. A tentative interpretation for this fact is that, even with $p_{min} = 0$, the reward estimates do not converge to 0, even for long epochs ($\Delta T = 200$) and does not prevent the worst operators from catching up later.

In the meanwhile, *Probability Matching* and *Adaptive Pursuit* display a lesser stability (Figure 2), with a much smaller area of high TCR compared to MAB algorithms. Again, the surface responses obtained for uniform and boolean scenarios, and for *Probability Matching* and *Adaptive Pursuit*, are similar.

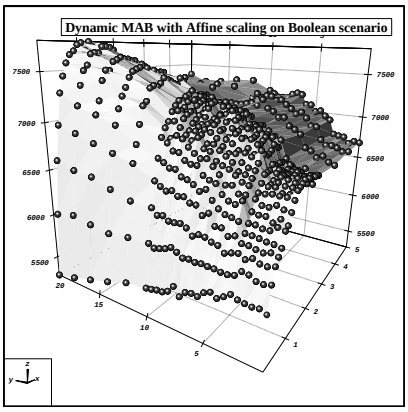


Figure 1: Response surface for a subset of the DOE for the D-MAB-A algorithm in the *Boolean* scenario for $\Delta T = 200$. Depth is the scaling factor $\mathcal{S}$and left to right is the threshold λ.

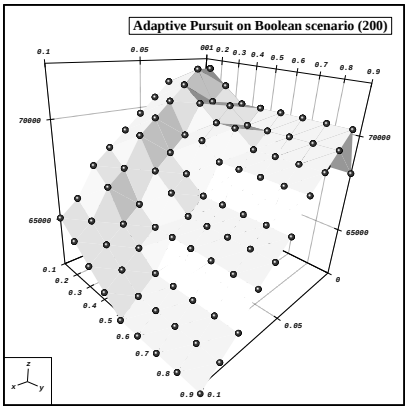


Figure 2: Response surface for a subset of the DOE for the Adaptive Pursuit algorithm in the *Boolean* scenario for $\Delta T = 200$. Depth is P_{min} and left to right is the adaptation rate α.

6.3 Comparison of the algorithms

Based on the extensive experimental evidence summarized in Tables 2 and 3, the TCR performances of all six algorithms are compared using a 1-way ANOVA test (confidence level 95%). This test is used to determine whether the performances of the algorithms are significantly different; it this

	PM P_{min} / α	AP P_{min} / α / β	MAB-M $\mathcal{S}$	MAB-A $\mathcal{S}$	D-MAB-M $\mathcal{S}$/ λ	D-MAB-A $\mathcal{S}$/ λ
Param.	**0**-.01 / .8-**.9**	**.03** / .7-**.9** / .4-**.9**	.3-**.4**-.5	**.5**-1.4	.6-**.7**-.8 / **4**-13	1.2-**1.5**-1.9 / 1.5-**2**-4
TCR	1.80 ± 0.077	2.12 ± 0.12	2.25 ± 0.033	1.98 ± 0.047	2.24 ± 0.031	2.20 ± 0.044
$\hat{p}_{\text{best}}$	0.32 ± 0.06	0.58 ± 0.09	0.73 ± 0.04	0.45 ± 0.032	0.68 ± 0.038	0.66 ± 0.07
Param.	0-**.01** / .9	**.03**-.1 / .7-**.9** / .7-**.9**	**.2**-.6	.4-**1.9**	.5-**.7**-1 / 5-**21**	.8-**1.1**-1.8 / **17** (all)
TCR	1.82 ± 0.29	1.84 ± 0.29	1.89 ± 0.135	1.74 ± 0.132	1.85 ± 0.147	1.81 ± 0.145
$\hat{p}_{\text{best}}$	0.32 ± 0.15	0.33 ± 0.13	0.36 ± 0.05	0.30 ± 0.05	0.35 ± 0.07	0.33 ± 0.07
Param.	$\emptyset$ / .1	.02-**.09** / **.1**-.4 / $\emptyset$	**.1** (all)	**.4** (all)	**.2**-.6 / $\emptyset$	**.2**-.4 / $\emptyset$
TCR	1.60 ± 0.49	1.57 ± 0.49	1.63 ± 0.24	1.59 ± 0.27	1.58 ± 0.234	1.58 ± 0.219
$\hat{p}_{\text{best}}$	(0.22 ± 0.07)	0.22 ± 0.09	0.24 ± 0.06	0.22 ± 0.05	0.22 ± 0.02	0.21 ± 0.004

Table 2: Results for $\Delta T = 50$. Each big row is a scenario (from top to bottom, Uniform, Boolean and Outlier). Each column presents the optimal parameter setting and the corresponding results of one algorithm.

	PM P_{min} / α	AP P_{min} / α / β	MAB-M $\mathcal{S}$	MAB-A $\mathcal{S}$	D-MAB-M $\mathcal{S}$/ λ	D-MAB-A $\mathcal{S}$/ λ
Param.	**0** / **.8**	**.02** / .7-**.8** / .3-**.7**-.9	.3-**.4**	**.4**-1.1	.8-**1.2**-1.6 / **4.5**-12	3-**5** / 3-**8**-14
TCR	7.36 ± 0.21	9.19 ± 0.25	9.46 ± 0.111	8.18 ± 0.058	9.75 ± 0.063	9.79 ± 0.059
$\hat{p}_{\text{best}}$	0.34 ± 0.02	0.75 ± 0.06	0.84 ± 0.009	0.47 ± 0.017	0.92 ± 0.014	0.94 ± 0.010
Param.	0-**.01** / .5-**.9**	.01-**.02** / .5-**.9** / .2-**.9**	.1-**.2**-.4	**.4**-1.4	.9-**1.5**-2 / 4.5-**12**	1.1-**1.5**-2 / 14-**16**-18
TCR	7.90 ± 0.73	8.13 ± 0.38	8.19 ± 0.341	7.50 ± 0.360	7.57 ± 0.421	7.71 ± 0.347
$\hat{p}_{\text{best}}$	0.43 ± 0.14	0.48 ± 0.08	0.52 ± 0.05	0.35 ± 0.07	0.38 ± 0.05	0.41 ± 0.06
Param.	$\emptyset$ / **.1**	.04-**.07** / .1 / $\emptyset$	.1-**.2**-.4	.1-**.3**-1.8	.1-**.2**-.5 / $\emptyset$	.2-**.3**-.5 / $\emptyset$
TCR	6.50 ± 1.12	6.66 ± 1.24	7.12 ± 0.585	6.83 ± 0.634	6.23 ± 0.515	6.17 ± 0.435
$\hat{p}_{\text{best}}$	0.24 ± 0.07	0.27 ± 0.04	0.29 ± 0.07	0.26 ± 0.07	0.22 ± 0.01	0.21 ± 0.006

Table 3: Results for $\Delta T = 200$. Each big row is a scenario (from top to bottom, Uniform, Boolean and Outlier). Each column presents the optimal parameter setting and the corresponding results of one algorithm.

is the case, a Scheffé-S[4] test is run to extract the best algorithm.

After these results, the algorithms based on the Multi-Armed Bandit framework outperform the other methods. For the *Uniform* scenario with frequent changes ($\Delta T = 50$), MAB-M and D-MAB-M (MAB with Multiplicative Scaling, static or dynamic) outperform all other methods. When considering longer epochs ($\Delta T = 200$), D-MAB-M and D-MAB-A (the dynamic multi-armed bandits using whatever scaling method) are statistically better than the other methods. Figure 3 (a) and (b) illustrates this situation with the usual box-plots.

For the Boolean scenario, MAB-M (the static MAB with multiplicative scaling) outperforms all other algorithms (Figure 3 (c)), for both short and long epochs.

7. CONCLUSION

This paper has introduced D-MAB, a new Adaptation rule (or Decision Making method) to work in dynamic environments, hybridizing the Multi-Armed Bandit algorithm UCB with the Page-Hinkley test to detect abrupt changes in the environment. D-MAB has been compared to the state-of-the-art methods of *Probability Matching* and *Adaptive Pursuit* on artificial scenarios, and performed very well on the original scenario proposed in [20]. Surprisingly, the simple Multi-Armed Bandit algorithm was found adaptive enough (given the speed of the changes) to perform even better in another scenario where the reward is boolean instead of uniformly distributed.

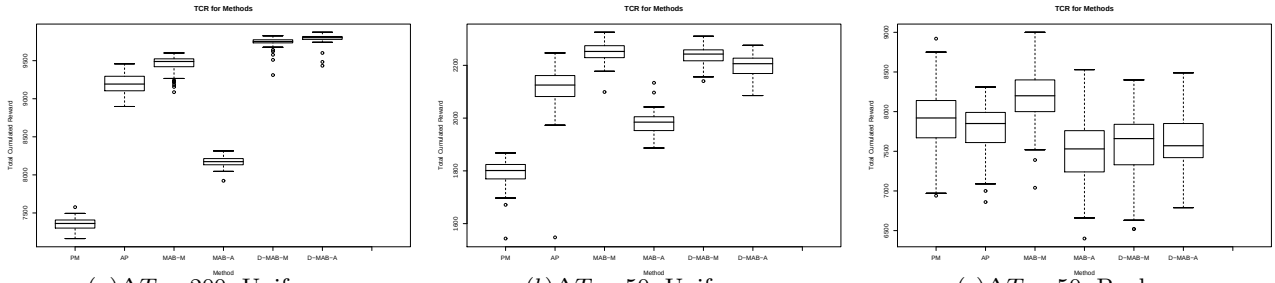
While D-MAB involves 2 additional parameters, the extensive experimental study presented in this paper suggests that it is rather robust with respect to their settings, and comparatively more robust than *Probability Matching* and *Adaptive Pursuit*. Further work is concerned with automatically setting the D-MAB parameters.

Another research perspective is concerned with addressing the operator adaptation within the evolutionary run. The experiments proposed here, though probably far from reality, can nevertheless give us some hints when it comes to design credit assignment mechanisms.

The results for the Probability Matching and the Adaptive Pursuit algorithms are surprising: whereas they confirm those of [20] for the *Uniform* scenario, their results are not statistically different for the other two scenarios. Moreover, the best value for the minimal probability for all operators is much lower than the one used in [20]. This seems to indicate that, at least in the proposed scenarios, the changes are occurring too rapidly to allow probabilities to go to 0, and greediness is a valid option.

The case of the *Outlier* scenario was poorly handled by all methods. However, it has been acknowledged that the ability for an operator to produce outliers could be a good measure of their efficiency within an Evolutionary Algorithm [23]. Even though this scenario was a caricature, it should be addressed by an efficient Adaptive Operator Selection method.

[4]The S method allows for comparison between the means of populations, either from different size or not. It is a very robust method in terms of normality of the population. For more details see [3, p. 224-225]



$(a)\Delta T = 200$, Uniform $(b)\Delta T = 50$, Uniform $(c)\Delta T = 50$, Boolean

Figure 3: Box-plots of the six algorithms for the scenarios where visible differences exist. In each plot, from left to right, PM, AP, MAB-M, MAB-A, D-MAB-M, and D-MAB-A.

8. ACKNOWLEDGMENTS

The authors thank Olivier Teytaud for fruitful discussions, and gratefully acknowledge the support of the Network of Excellence PASCAL, IST-2002-506778.

9. REFERENCES

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

[2] H. J. C. Barbosa and A. M. Sá. On adaptive operator probabilities in real coded genetic algorithms. In *Proc. XX International Conference of the Chilean Computer Science Society*, 2000.

[3] R. Bertrand. *Pratique de l'analyse statistique des donnees*. Presses de l'Universite du Quebec, Quebec, QC, 1986.

[4] L. Davis. Adapting operator probabilities in genetic algorithms. In *Proc. 3rd International Conference on Genetic Algorithms*, pages 61–69. Morgan Kaufman, 1989.

[5] K. De Jong. Parameter Setting in EAs: a 30 Year Perspective. In F. Lobo, C. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, chapter 1, pages 1–18. Springer Verlag, 2007.

[6] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in Evolutionary Algorithms. *IEEE Trans. Evolutionary Computation*, 3(2):124, 1999.

[7] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter Control in Evolutionary Algorithms. In F. Lobo, C. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, chapter 2, pages 19–46. Springer Verlag, 2007.

[8] D. Goldberg. Probability Matching, the Magnitude of Reinforcement, and Classifier System Bidding. *Machine Learning*, 5(4):407–426, 1990.

[9] J. Grefenstette. Optimization of Control Parameters for Genetic Algorithms. *IEEE Trans. Systems, Man and Cybernetics*, 16:122–128, 1986.

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

[11] Z. Hussain, P. Auer, N. Cesa-Bianchi, L. Newnham, and J. Shawe-Taylor. Exploration vs. exploitation challenge. In *http://www.pascal-network.org/Challenges/EEC/*, 2006.

[12] B. A. Julstrom. What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm on genetic algorithms. In *Proc. 6th International Conference on Genetic Algorithms*, pages 81–87. Morgan Kaufmann, 1995.

[13] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6:4–22, 1985.

[14] F. Lobo and D. Goldberg. Decision making in a hybrid genetic algorithm. In *Proc. IEEE International Conference on Evolutionary Computation*, pages 121–125. IEEE Press, 1997.

[15] F. Lobo, C. Lima, and Z. Michalewicz, editors. *Parameter Setting in Evolutionary Algorithms*. Springer, 2007.

[16] G. A. Moore. *Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customer*. Collins Business Essentials, 1991.

[17] E. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.

[18] G. Piriou, F. Coldefy, P. Bouthemy, and J.-F. Yao. Détection supervisée d'événements à l'aide d'une modélisation probabiliste du mouvement perçu. In *Proc. 14ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle*, 2004.

[19] W. Spears. Adapting crossover in evolutionary algorithms. In *Proc. 4th Annual Conference on Evolutionary Programming*, pages 367–384. MIT Press, Cambridge, MA, 1995.

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

[21] D. Thierens. Adaptive Strategies for Operator Allocation. In F. Lobo, C. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, pages 77–90. Springer Verlag, 2007.

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

[23] J. M. Whitacre, T. Q. Pham, and R. A. Sarker. Use of statistical outlier detection method in adaptive evolutionary algorithms. In *Proc. Genetic and Evolutionary Computation Conference*, pages 1345–1352. ACM, 2006.