Extreme Compass and Dynamic Multi-Armed Bandits for Adaptive Operator Selection

Jorge Maturana, Álvaro Fialho, Frédéric Saubion, Marc Schoenauer, Michèle Sebag

***Abstract*— The goal of *Adaptive Operator Selection* is the on-line control of the choice of variation operators within Evolutionary Algorithms. The control process is based on two main components, the *credit assignment*, that defines the *reward* that will be used to evaluate the quality of an operator after it has been applied, and the *operator selection* mechanism, that selects one operator based on some operators qualities. Two previously developed *Adaptive Operator Selection* methods are combined here: *Compass* evaluates the performance of operators by considering not only the fitness improvements from parent to offspring, but also the way they modify the diversity of the population, and their execution time; *Dynamic Multi-Armed Bandit* proposes a selection strategy based on the well-known UCB algorithm, achieving a compromise between exploitation and exploration, while nevertheless quickly adapting to changes. Tests with the proposed method, called *ExCoDyMAB*, are carried out using several hard instances of the Satisfiability problem (SAT). Results show the good synergetic effect of combining both approaches.**

I. Introduction

Evolutionary Algorithms (EA) constitute efficient solving methods for general optimization problems. Their performances have been assessed on a wide range of applications. From an operational point of view, they can be considered as a basic computational process that selects and applies transformation operators on sets of possible configurations of the problem to be solved.

Here, we distinguish the general structure of the algorithm, which includes the management of the population (i.e., size setting, selection process, etc), from its configurable components. The design of the components of the algorithm consists in choosing a suitable encoding of the search space and in defining the required variation operators, namely the mutation and recombination operators. However, once these operators have been chosen, a difficult and crucial task remains: how to control the general computation process? This control is usually embedded in a set of parameters that can be related to the data structures or to computation steps.

Therefore, parameter setting in EA appears as a major issue that has deserved much attention during recent years [1]. Parameter setting may indeed be considered at two complementary levels:

J. Maturana and F. Saubion are with the LERIA laboratory, Université d'Angers, Angers, France (email: {*LastName*}*@info.univ-angers.fr*).

A. Fialho, M. Schoenauer and M. Sebag are with the Microsoft Research – INRIA Joint Centre, Orsay, France (email: {*FirstName.LastName*}*@inria.fr*). M. Schoenauer and M. Sebag are also with the Project-Team TAO, INRIA Saclay – Île-de-France & LRI (UMR CNRS 8623), Orsay, France.

Design: In many application domains, that directly pertain to standard representations, users who are not EC-experts can simply use off-the-shelf EAs, using classic (and thus non specialized) variation operators to solve their problems. However, the same users will encounter great difficulties when facing problems that fall out of the basic frameworks. The design of problem-specific operators or encodings requires much expertise, though some advance tools are now available [2]. In any case, the impact on the computation process of problem-specific operators is even more difficult to forecast than those of well-known operators, and thus their associated parameters are harder to set. In all cases, there is a need for automatic parameter tuning for as many parameters as possible.

Performance: Once the general architecture of the algorithm has been defined, the user needs to configure the behavior of these components through parameters. This setting has a direct impact on the algorithm's performance and reliability. Indeed, this efficiency is strongly related to the way the Exploration versus Exploitation (EvE) dilemma is addressed, determining the ability of the EA to escape from local optima in order to sparsely visit interesting areas and reach global solutions. According to the taxonomy of [3], we should distinguish between parameter tuning, which occurs before the run, by means of extensive and often costly collections of preliminary experiments; and parameter control, that takes place during the run of the algorithm, by using some external knowledge or method.

From these observations, a current trend in EA is to focus on the definition of more autonomous solving processes, which aim at allowing the basic user to benefit from a more efficient and easy-to-use algorithmic framework. This approach is actually not limited to EA, being also investigated in the operation research and constraint programming fields, where the current amount of solving technologies that are included in efficient solvers require a huge expertise to be fully used [4]. Note that though the control generally handles the parameters, it can also act on the components themselves. For instance, hyper-heuristics [5] can be viewed as general control methods that select suitable solving techniques before or during the run, according to a higher level algorithm.

This work focuses on the control of the parameters related to the variation operators within an EA, namely mutation, recombination and local search operators, more precisely on the on-line control of operators application rates during the run. Indeed, the design of autonomous or, at least, more self-controlled EAs should involve control mechanisms that take

into account two important features: learning, which allows the algorithms to acquire knowledge before or along the solving process; and adaptation, that uses what was learned to modify the configuration of the algorithm. These two complementary aspects can be handled by a combination of the following two mechanisms:

- *Credit Assignment*: Learning about the relative efficiency of the operators is usually achieved by giving them some *reward* after they have been applied. Defining such rewards involves many critical choices.
- *Operator Selection*: The choice of an operator, based on the series of rewards distributed in the past, is made using a *selection rule* that needs to be carefully designed and parametrized.

The authors have recently and separately developed two *Adaptive Operator Selection* systems that share the previous principles, but differ on their practical integration in EA.

In [6], *Credit Assignment* is done by means of a mechanism that assess the efficiency of each operator according to several criteria: diversity of the population, average fitness and execution time. The EvE dilemma can be easily tackled by setting a search direction in terms of diversity and quality. The rewards are thus computed according to an average measure of the criteria for the operators w.r.t. that search direction. *Operator Selection* is achieved by adapting operators application rates according to a proportional probability scheme, the so-called *Probability Matching* [7], even though it is not explicitly referred to in the paper. For this reason, in the present work, *Compass* will refer to the *Credit Assignment* method proposed in [6], and not to the entire *Adaptive Operator Selection*.

In [8], operators efficiency is assessed through the fitness improvement of the newborn offspring compared to its parent. However, the rewards are not based on mean values, but rather focused on extreme fitness improvement values, in order to detect operators that have a great impact at a particular computation step and try to favor them for a while. The *Operator Selection* mechanism is the so-called *Dynamic Multi-Armed Bandit* (DMAB). Initially introduced in [9], this is a technique adapted from the Game Theory field, that optimally exploits the current empirical best operator, while nevertheless keeping on trying the other ones. The whole mechanism is referred as *Ex-DMAB*.

The main weakness of the former *Adaptive Operator Selection* system lies in its *Operator Selection* mechanism, which is rather simple, thus application probabilities change slow and conservatively; the latter system lacks a better *Credit Assignment*, relying only on fitness improvement, without taking into account the population diversity that seems mandatory to tackle multi-modal problems. It seems therefore a good idea to combine both methods, hopefully getting the best of both worlds.

This work presents an empirical analysis of such combination, called *ExCoDyMAB*, showing its efficiency in comparison with the original methods *Compass* and *Ex-DMAB*. The framework is applied on a number of instances of the satisfiability problem, with results showing that we are in the right way towards an efficient and more autonomous framework for doing *Adaptive Operator Selection*.

This paper is organized as follows: Section II review past publications on the subject of *Adaptive Operator Selection*, Section III recalls the two approaches previously developed by the authors that are combined in this work, as presented in Section IV. Section V defines the experimental testbed, while Section VI presents how the architecture and the meta-parameters of our controller were defined. Finally, Section VII presents the experimental results, and some conclusions and perspectives of future work are drawn in Section VIII.

II. Adaptive Operator Selection

Adaptive methods use information from the history of evolution to modify parameters while solving the problem. This paper focuses on the *Adaptive Operator Selection* (AOS), i.e., the definition of an on-line strategy able to autonomously select between different variation operators. Fig. 1 illustrates the general scheme for achieving this goal, from which we can derive the need of defining two main components: the *Credit Assignment* - how to assess the performance of each operator based on the impact of its application on the progress of the search; and the *Operator Selection* rule - how to select between the different operators based on the rewards that they have received so far. Given the separability of these two issues, this section will survey existing AOS methods, looking at how they addressed them.

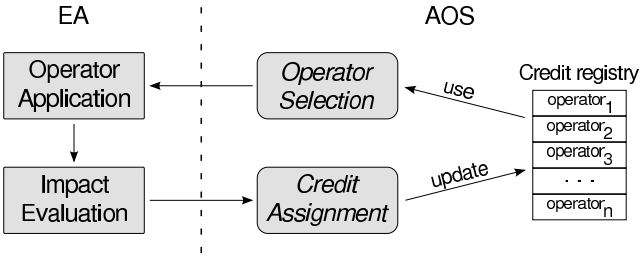

Fig. 1. General *Adaptive Operator Selection* scheme.

A. Learning by Credit Assignment

Since the first works in AOS in the late 80's [10], several methods to assign credit (or reward) to variation operators have been proposed in the literature. The difference among them lies in how they measure the quality of an operator according to the result of its application.

Most methods use only the fitness improvement as a reward, comparing the quality of the newborn offspring with those of its parents ([11], [12], [13]), the current best [10], or the median [14] individual. If no improvement is achieved, usually a null credit is assigned to the operator. Admittedly, no clear conclusive result can be gathered from these works.

More recently, in [15], a more sophisticated quality assessment was proposed, that aims at detecting outliers in the fitness distribution. Reported comparative results with other credit assignment techniques are conclusive, indicating

the superiority of this approach over a set of continuous benchmark problems. Though calling to another measure, the method proposed here borrows the idea of detecting beneficial but rare events.

Besides "what" is measured, another feature that needs to be defined is "who" should be measured. Some authors assign credit to the operators used to generate the ancestors of the current individual (e.g. using some bucket brigade algorithm [10], [14]), claiming that the creation of efficient parents is indeed as important as the creation of improved offspring. Others, however, do not consider ancestors ([11], [12]) and some even suggest that it sometimes degrades the results [13].

B. Adaptation by Operator Selection Rule

The most straightforward (and so the most common) way of doing *Operator Selection* is the *Probability Matching* (PM) ([7], [11], [13]). Basically, a roulette wheel-like process selects the operator to be applied with a probability proportional to its empirical quality (e.g., the average of received rewards). A minimal probability (p_{min}) is usually applied, to enforce a minimal amount of exploration, preventing the "loss" of any operator that might become the good one at some further stage of the search. Clearly, if some operator receives no reward (respectively, the maximal reward) for some time, its expected reward will go to p_{min} (respectively, $p_{max} = 1 - K * p_{min}$). However, this convergence is very slow, and, experimentally, all mildly relevant operators keep being selected, thus hindering their performance [16].

This drawback is partly addressed by the *Adaptive Pursuit* (AP) [16], a method originally proposed for learning automata, that implements a winner-take-all strategy. A user-defined parameter β controls the greediness of the strategy, i.e., how fast the probability of selecting the current best will converge to p_{max} while all the others will go to p_{min}.

Others, like APGAIN [17], propose an *Operator Selection* divided in two periods. During the first stage, operators are randomly selected, and the rewards received are used to build some knowledge about them. In a second stage, the operator to be applied is selected according to the quality assessed during the learning phase. These two phases are repeated alternatively, so the changes can be caught up. The problem is that, roughly, a quarter of the generations is dedicated to the first (learning) period, i.e., random selection, what may strongly affect the performance of the algorithm in case of using disrupting operators.

III. Compass and *Ex-DMAB*

In this section, we describe the two AOS methods independently developed by the authors ([6], [8]). The *Credit Assignment* mechanisms and the *Operator Selection* rules proposed in each work are presented, showing their complementarity, which is the motivation of the present work.

A. Compass

The main contribution of the work in [6] is its *Credit Assignment* mechanism, which works as follows. Firstly, three measures are gathered each time an operator is applied: population diversity variation, mean fitness variation and execution time (Fig. 2.a). The average of the values of the sliding window that stores the last τ applications is displayed in a "diversity vs fitness" plot (black points in Fig. 2.b). A user-defined parameter Θ defines a compromise between obtaining good results and maintaining diversity in the population, addressing the EvE dilemma. In practice, such angle defines the plane according to which perpendicular distances from the points are measured. Finally, these measures are divided by the operator's execution time, obtaining the final aggregated evaluation of each operator (Fig. 2.c), which is the reward that will be used to update the selection preferences.

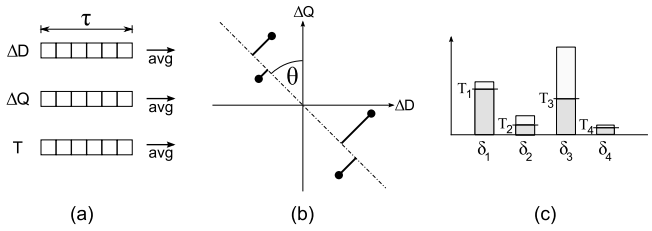

Fig. 2. *Compass* credit assignment: Sliding windows of three measures are maintained (a). Average measures of ΔD and ΔQ are plotted and distance of those points are measured according to a plane with a slope of Θ (b). Finally, those distances are divided by the execution time, resulting in the reward assigned to the operator.

Once the reward is calculated, *operator selection* is done by choosing the operator to apply with a probability proportional to the obtained rewards (*Probability Matching*).

This approach was used to control a genetic algorithm that aims at solving the well-known Boolean satisfiability problem (SAT), automatically selecting between 6 ill-known operators. The SAT problem was chosen because it offers a large variety of instances with different properties and search landscapes, besides allowing the scaling of the instance difficulty. The experiments have demonstrated that this *AOS* method is efficient and provides good results w.r.t. other existing mechanisms, such as AP [16] and APGAIN [17] based on fitness improvement.

It must be noted that such results were achieved mainly due to the strength of the credit assignment mechanism proposed, which provides a robust measurement of the impact of the operator application by simultaneously considering several criteria. As already discussed in Section II-B, the operator selection rule used (PM) is rather simple, known as being quite conservative and slow w.r.t. the adaptation.

B. Ex-DMAB

The two ingredients of the *Adaptive Operator Selection* method proposed in [8] are a *Credit Assignment* based on extreme values of fitness improvement, and an *Operator Selection* rule based on the Multi-Armed Bandit paradigm.

The idea of extreme fitness improvement was proposed as the *Credit Assignment* mechanism based on the assumption that attention should be paid to extreme, rather than average, events. This is in agreement with [15], and is implemented by

measuring the difference of fitness between the parent and the offspring generated by the operator application, and assigning as reward the maximum of such fitness improvements over a sliding window of the last κ applications.

Concerning the *Operator Selection* rule, the explored idea, first proposed in [9], is that the selection of an operator can be seen as yet another Exploration vs. Exploitation dilemma, but this time at operator-selection level: there is the need of applying as much as possible the operator known to have brought the best results so far, while nevertheless exploring the other possibilities, in case one of the other operators becomes the best option at some point. Such dilemma has been intensively studied in the context of *Game Theory*, in the so-called Multi-Armed Bandit (MAB) framework. Among the existent MAB variants, the *Upper Confidence Bound* (UCB) [18] was chosen to be used, for being proved optimal w.r.t. maximization of the cumulative reward.

More formally, the UCB algorithm works as follows. Each variation operator is viewed as an *arm* of an MAB problem. 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 empirical reward received until time t by arm i. At each time step t, the algorithm selects the arm maximizing the following quantity:

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

The first term of this equation favors the best empirical arm (exploitation) while the second term ensures that each arm is selected infinitely often (exploration); this algorithm has also been described briefly as "be optimistic when facing the unknown", as the second term of Equation 1 can also be viewed as some kind of variance, and the user should choose the arm that might lead to the highest value. In the original setting [18], all rewards, and hence also their empirical means $\hat{p}_{i,t}$ were in $[0, 1]$. However, because this is not the case here, a *Scaling factor* C is needed, in order to properly balance the trade-off between both terms.

Another important issue is that the original MAB setting is static, while the AOS scenario is dynamic, i.e., the quality of the operators is likely to change along the different stages of the search. Even though the exploration term in the UCB algorithm ensures that all operators will be tried infinitely many times, after a change in the ordering of operators, it might take a long time before the new best operator catches up. To cope with dynamic environments, it has been proposed [19] to use a statistical test that efficiently detects changes in time series, the *Page-Hinkley* (PH) test [20], coupled with the UCB algorithm. Basically, as soon as this test detects a change in the distribution (e.g. the "best" operator is probably not the best anymore), the MAB algorithm is restarted, allowing it to quickly re-discover the new best operator.

More precisely, the PH test works as follows: let $\bar{r}_t$ denote the average of $r_1, \ldots r_t$ and let e_t denote the difference $r_t - \bar{r}_t + \delta$, where δ is a tolerance parameter. The PH test considers the random variable $m_t = \sum_1^t e_i$. When the difference between $M_t \equiv \max_{i \leq t} m_i$ and m_t is greater than some user-specified threshold γ, the PH test triggers.

The MAB part in *Ex-DMAB* involves one meta-parameter, the scaling factor C, while the PH test involves two parameters, the detection's threshold γ, and δ, which enforces the robustness of the test when dealing with slowly varying environments. Note that, according to initial experiments in [9], δ has been kept fixed to 0.15. The dynamic MAB algorithm has been termed *DMAB*, with the complete *AOS* combination being denoted *Ex-DMAB*.

Ex-DMAB has been used to adapt a (1+λ)-EA, by efficiently choosing on-line between 5 mutation operators for solving the OneMax problem [9], and has been tried on yet another unimodal benchmark problems, the long k-path [21]. Experiments showed that the use of extremes rather than averages is really beneficial. Besides, this AOS technique performed similarly to *Adaptive Pursuit* (also using extreme fitness improvement as reward), achieving a performance not far from the known optimal deterministic strategy (found by means of Monte-Carlo simulations).

However, as it only takes into account the fitness improvement, such *AOS* method would probably be less efficient on rougher landscapes, quickly converging to local optima. For this reason, diversity should also somehow be considered. Another drawback of *Ex-DMAB* is that, since the variance of fitness improvements changes as the search advances, there does not exist any value for C and γ parameters that are likely to perform well during the whole search. Possible solutions to this drawbacks are either to also adapt those meta-parameters, or to use a credit assignment that keeps the same variance w.r.t. the rewards.

IV. *ExCoDyMAB*= *Compass* + *Ex-DMAB*

Previous Section showed that the strengths and weaknesses of both *Compass* and *Ex-DMAB* methods are complementary: *Compass* measures in a holistic way the effect of operators application over the population, but the operator selection rule is rather rudimentary, while *DMAB* has an effective way to adapt and select the operators, but its credit assignment mechanism is probably too simplistic. It seems hence natural to combine both methods. However, even though merging both modules is straightforward, some important issues need to be further explored:

- *Compass* uses sliding windows in the "measurement stage", with a unique reward value in its output; while *Ex-DMAB* stores in a sliding window the last κ outputs (rewards) of the *Credit Assignment* module. Should we keep both windows, or would it be redundant? And if only one is kept, which one should it be? From here on, these two windows will be referred to as $W1$ for the measurement window and $W2$ for the reward window.
- Another issue concerning the sliding windows is that of their usage: should the algorithm use their average, extreme, or simply instantaneous value? (equivalent to using no window at all). The *Extreme* was shown to be stronger in a unimodal problem [8], but how does such results hold in this completely different scenario?

- The last issue concerns the other meta-parameters. Besides the size and type of $W1$ and $W2$, we need to tune the values of the angle Θ in *Compass*, and the scaling factor C and change detection threshold γ in *DMAB*. Since the idea is not to replace some parameters (the operator application probabilities) by other ones, even at a higher level of abstraction, we need to better understand their effects. One way to do so is to experimentally study their influence on the performance of the *AOS* in situation, and propose some robust default values.

An empirical analysis of such issues will be presented in the following.

V. Experimental setting

ExCoDyMAB has been experimented within the same Evolutionary Algorithm that has been used in [6]. This algorithm solves the well-known combinatorial SAT problem [22], which consists in assigning values to binary variables in order to satisfy a Boolean formula.

An instance of the SAT problem is formally defined by a set of Boolean variables $\mathcal{X} = \{x_1, \ldots, x_n\}$ and a Boolean formula $\mathcal{F}$: $\{0,1\}^n \rightarrow \{0,1\}$. The formula is said to be satisfiable if there exists an assignment v: $\mathcal{X} \rightarrow \{0,1\}^n$ satisfying $\mathcal{F}$, unsatisfiable otherwise. Instances are classically formulated in conjunctive normal form (conjunctions of clauses) and one thus has to satisfy all these clauses. Given that SAT was the first problem to be proved NP-complete, many different problems from both real world and theoretical background have been expressed as SAT instances. So, by tackling such problem, we can deal with a diverse set of fitness landscapes with different characteristics.

For the present work, we have extended the number of SAT instances used in [6] (originally obtained from [23]), by adding some harder instances from the SAT 2006 Race. Table I shows the instances used here, pointing out whether they are satisfiable or not, their family, and the number of variables and clauses they involve. All 22 instances have been considered for the final comparison, but only 7 of them were taken into account during the definition of the platform and (meta-)parameter analysis presented in Section VI. This subset, marked with an asterisk in Table I, was chosen among the hardest instances with short enough running times, reducing the experimental cost for the platform definition. Tuning the meta-parameters on a small set of instances and testing them further on "unseen" instances witnesses the generality of the tuned parameters.

The EA uses a standard binary representation (one bit per boolean variable), and 6 non-standard variation operators, that are applied within a steady-state algorithm. As in [6], the purpose here is not to use state of the art SAT operators, but rather to manage a set of completely unknown operators, as a naive user would do when facing a new problem. Desirably, the *AOS* mechanism should then be able to autonomously discriminate good from bad operators, at any given time of the search. The operators are now briefly recalled:

TABLE I
SAT instances used

Problem	Sat?	# Vars.	# Clauses	Family
4blocks	Yes	758	47820	Blocks World Problem
aim	Yes	200	320	Random-3-SAT
f1000	Yes	1000	4250	Random-3-SAT
CBS	Yes	100	449	Controlled Backbone
Flat200	Yes	600	2237	Flat Graph Coloring
logistics	Yes	828	6718	Logistics Planning
medium	Yes	116	953	Randomly Generated
Par16	Yes	1015	3310	Parity Learning Problem
sw100-p0	Yes	500	3100	Morphed Graph Coloring
sw100-p1	Yes	500	3100	Morphed Graph Coloring
Uf250	Yes	250	1065	Phase Transition Region
Uuf250	No	250	1065	Phase Transition Region
Color*	No	1444	119491	Chessboard Coloring
G125*	Yes	2125	66272	Graph Coloring
Goldb-heqc*	No	5980	35229	Randomly Generated
Grieu-vmpc	Yes	729	96849	Randomly Generated
Hoons-vbmc*	No	8503	25116	Randomly Generated
Schup	No	14809	48483	Randomly Generated
Simon*	No	2424	14812	Randomly Generated
Manol-pipe	Yes	14052	41596	Pipelined Machine Verification
Velev-eng*	No	6944	66654	Pipelined Machine Verification
Velev-sss*	No	1453	12531	Pipelined Machine Verification

- *One-point Crossover* randomly chooses two individuals and a random position, and exchanges their first and second parts. The best child then replaces the worst parent (no global replacement here).
- *Contagion* randomly chooses two individuals and sets the variables in all false clauses of the worst individual to the values they have in the best one.
- *Hill Climbing* checks all neighbors at Hamming distance 1 and moves to the best one, repeating the process as long as it improves the fitness. It is a local search operator, but has been included here for the sake of diversity of variation operators.
- *Tunneling* swaps variables without decreasing the number of true clauses, according to a tabu list of length equal to $\frac{1}{4}$ of the number of variables (it can be seen, again, as a local search operator).
- *Bad Swap* swaps all variables that appear in false clauses, whatever their values are.
- *Wave* swaps the values of the variable that appears in the highest number of false clauses and in the minimum number of clauses only supported by it; the process is repeated at most $\frac{1}{2}$ times the number of variables.

The population size (3) and maximum number of generations (5000 – the only stopping criterion) were arbitrary fixed. On-going work is concerned with checking that the results presented in Section VII still holds in different configurations of these parameters.

VI. Architecture definition and meta-tuning

We must define how to efficiently integrate *Compass* and *Ex-DMAB*. The first decision concerns whether to include or not the sliding windows $W1$ and/or $W2$, and which should be their outputs. The following possible output policies are available (for each window):

- *Average* (A) value from stored measures;
- *Extreme* (E) value (maximum) from stored values (except for the execution time, kept in $W1$, which does not make sense);
- *Instantaneous* (I) value, i.e., no sliding window.

Besides, the following meta-parameters also need to be analyzed and tuned:

- The size of $W1$, τ and the size of $W2$, κ.
- The *Compass* angle Θ, that defines the tradeoff between the EA exploration and exploitation.
- The *DMAB*'s scaling C parameter, that defines the tradeoff between exploration and exploitation at the operator-selection level.
- The *DMAB*'s γ parameter, the threshold of the change detection test that triggers the restarts.

Given the high number of possible configurations and the initial lack of knowledge about which should be the good ones, an off-line parameter tuning method was used to facilitate this empirical analysis: the F-Race ([24], [25]), a variant of the Racing methods that uses the Friedman's two-way Analysis of Variances by Ranks as statistical test to eliminate the candidates. Racing is an alternative to a full factorial Design of Experiment, in which M runs should be performed on N instances for each candidate solution: in Racing, candidate configurations are eliminated as soon as there is enough statistical evidence that they are worst than the current best one. Cycles of "execution/comparison/elimination" are repeated until there is one candidate left, or some other stopping criteria is achieved. So, instead of wasting computing time to estimate with precision the performance of inferior candidates, it allocates the computational resources in a better way, by focusing on the most promising ones and consequently achieving lower variance estimates for them. Racing typically requires 10-20% of the computing time of a complete factorial DOE.

The domains for the different parameters are as follows: $C \in \{5, 7, 10\}$; $\gamma \in \{1, 3, 5\}$; and the windows type(size) combinations $\in \{A(10), A(50), E(10), E(50), I(1)\}$ for both $W1$ and $W2$. Thus, the initial number of possible configurations is 225.

The angle Θ for *Compass* was set to 0.25, as preliminary experiments have shown that a different value causes a positive-feedback phenomenon[1], that moves the EA to an extreme behavior. Figure 3 shows the fitness improvement when using different values of Θ. Note that all values below 0.25 tend to produce a similar erratic behavior, while values above 0.25 have a poor improvement rate.

The stopping criteria for the Racing was set to 80 runs over all the instances, with eliminations taking place after each run, starting from the 11^{th}. As recommended in [24], in order to enable a fair comparison, all the experiments of the same epoch were started with the same initial population (the "blocking design" concept).

[1]In systems theory, positive feedback is a process in which a system responds to a perturbation in the same sense of the perturbation, thus distancing the system from its original state.

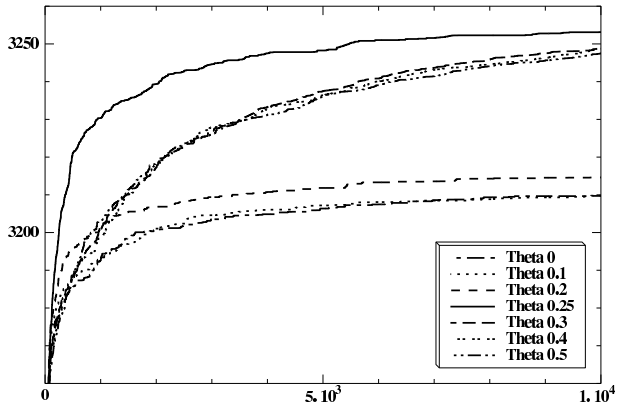

Fig. 3. Fitness improvement using different values of Θ

TABLE II
RACING SURVIVORS

Name	$W1$ type, size	$W2$ type, size	C	γ
A	Extreme, 10	Instantaneous	7	1
B	Extreme, 10	Average, 10	7	1
C	Extreme, 10	Average, 50	7	1
D	Extreme, 10	Extreme, 10	7	3

VII. RESULTS AND DISCUSSION

At the end of the Racing process, 4 configurations were still "alive", presented in Table II. This clearly indicates that the most important sliding window is $W1$, and it should be used in its Extreme configuration with a size of 10 (i.e. taking as *Compass* inputs the maximal of the last 10 values), not matter which kind/size of $W2$ is being used. This fact emphasizes the need to identify rare-but-good improvements, greatly supporting the idea raised in [8]. Besides, the size of 10 for $W1$ could be interpreted by the following reasoning. With the Extreme policy, a larger τ would produce a long perdurability of the extreme values, even when the behavior of the operator has already changed. In the other hand, a shorter value $\tau = 1$ (i.e., the same as choosing Instantaneous policy) would forget those "rare-but-good" cases. One could suppose that an optimal size for $W1$ depends on the fitness landscape and the operators used - further research is needed to better understand the setting of τ.

To check the generality of those parameters, 50 runs were performed on the 22 SAT instances with each of the 4 configurations, promoting a comparison between them, and also verifying their performance in relation to the baseline methods: the original combinations *Compass* and *Ex-DMAB* (including a Racing phase for *Ex-DMAB* similar to that of *ExCoDyMAB*), and the *Uniform Choice* of operators. The results of this comparison are presented in Table III. Each cell value represents the number of problems in which one architecture is significantly better than the other (using a Student T-test with 95% confidence). For example, in the lower left corner, "18-2" means that D outperformed *Compass* on 18 instances, while the opposite happened only

TABLE III

COMPARATIVE RESULTS ON THE 22 SAT INSTANCE: EACH CELL INDICATES THE NUMBER OF TIMES THE ROW-ALGORITHM IS BETTER THAN THE COLUMN ALGORITHM ACCORDING TO A STUDENT T-TEST WITH 95% CONFIDENCE.

	Compass	*Ex-DMAB*	*UC*	A	B	C	D	$\sum dom$
Compass		9-9	22-0	4-18	2-17	2-18	2-18	-39
Ex-DMAB	9-9		22-0	0-18	0-21	0-21	0-21	-59
UC	0-22	0-22		0-22	0-22	0-22	0-22	-132
A	18-4	18-0	22-0		0-1	0-5	0-2	46
B	17-2	21-0	22-0	1-0		0-2	3-1	59
C	18-2	21-0	22-0	5-0	2-0		4-0	70
D	18-2	21-0	22-0	2-0	1-3	0-4		55

2 times. Finally, the rightmost column shows the number of times that an architecture wins, minus the times that it loses, as a global measure of comparative quality.

The dominance of *ExCoDyMAB* is overwhelming, and confirms the hypothesis that motivated the combination of both *Compass* and *Dynamic Multi-Armed Bandit* approaches. These latter approaches alone, within their respective original combinations, present a performance roughly equivalent, and clearly inferior to the newly combined one – though outperforming in turn the Uniform Choice policy.

Another important point is the good generalization capacity of *ExCoDyMAB* within its meta-parameters - the best configurations found by Racing perform also very similarly when solving the unseen instances. This makes us think that a quite general setting of meta-parameters has been found, at least for the experimental testbed used in this work.

It is also worth noticing that the rewards obtained from *Compass* are now normalized according to the current highest one, i.e., the best operator, when selected, will receive a reward of 1. This has the beneficial effect of delivering rewards to *DMAB* that are always in the same range. Since the purpose of the meta-parameter C is to adjust the importance between successful and almost-forgotten operators, such normalized reward could make a value of C close to 7 a somewhat definitive value for different EAs and problem instances when using the proposed *AOS* combination. Further work will investigate this finding more deeply.

The meta-parameter γ is more difficult to grasp. The frequency of *DMAB* restarts seems to be related to the operators being used by the EA. Operators whose performance depends on the current zone of search space being explored should be reviewed more frequently than others that change less. Consider for instance the local search operator based on hill climbing: its performance, in terms of fitness improvement, will vary depending on whether the operated individual is on a slope or already at a local optima. On the other hand, a mutation operator will present a similar performance, in terms of diversity increasing, regardless the position of the operated individual. Moreover, the fitness landscape could have an effect on how γ affects the search control.

The results on the 22 SAT instances, using the best parameter configuration found for *ExCoDyMAB* (C), together with those of *Compass*, *Ex-DMAB* and *Uniform Choice* are shown in table IV. The columns show the mean number of false clauses after 5000 function evaluations, averaged over 50 runs, and the standard deviation between parentheses.

TABLE IV

COMPARATIVE RESULTS ON THE 22 SAT INSTANCES: AVERAGE (STD DEV.) NUMBER OF FALSE CLAUSES (OVER 50 RUNS)

Method Problem	*ExCoDyMAB* (C)	*Compass*	*Ex-DMAB*	*Uniform Choice*
4blocks	2.8 (0.9)	6 (0.9)	6.2 (0.9)	13.4 (0.6)
aim	1 (0)	1 (0)	1.2 (0.3)	3.6 (1.8)
f1000	10.3 (2.3)	30.9 (6.2)	16.4 (2.6)	55.8 (8.6)
CBS	0.6 (0.6)	0.4 (0.5)	1 (0.9)	7 (2.7)
Flat200	7.2 (1.7)	10.6 (2.1)	10.7 (2.2)	37.7 (5.5)
logistics	6.5 (1.3)	7.6 (0.5)	8.8 (1.5)	17.9 (4.1)
medium	1.5 (1.5)	0 (0)	1.8 (1.6)	8.8 (3.4)
Par16	15.2 (3.1)	64 (10.2)	24.1 (5.7)	131.1 (14.5)
sw100-p0	9.2 (1.2)	12.8 (1.4)	12.5 (1.7)	25.9 (3.4)
sw100-p1	0 (0)	0.5 (0.6)	1.1 (0.8)	11.3 (3.5)
Uf250	0.9 (0.7)	1.8 (0.9)	1.7 (0.8)	9.1 (3.3)
Uuf250	2.5 (1)	4.5 (1.2)	3.1 (1.1)	12.7 (3.2)
Color	48 (2.5)	61.3 (2.2)	49.3 (3.4)	80.4 (6.6)
G125	8.8 (1.3)	20.6 (2)	13.5 (1.7)	28.8 (4.6)
Goldb-heqc	72.9 (8.5)	112.2 (15.2)	133.2 (15.9)	609.7 (96.2)
Grieu-vmpc	16.7 (1.7)	15.2 (1.7)	19.6 (1.8)	24.1 (3.3)
Hoons-vbmc	69.7 (14.5)	268.1 (44.6)	248.3 (24.1)	784.5 (91.9)
Manol-pipe	163 (18.9)	389.6 (37.2)	321 (38.1)	1482.4 (181.5)
Schup	306.6 (26.9)	807.9 (81.8)	623.7 (48.5)	1639.5 (169.9)
Simon	29.6 (3.3)	43.5 (2.7)	35.3 (6.3)	72.6 (11.3)
Velev-eng	18.3 (5.2)	29.5 (7.3)	118 (37.1)	394 (75.8)
Velev-sss	2 (0.6)	4.6 (1)	5.9 (3.9)	62.7 (25.2)

Note that, as mentioned before, the purpose of this work was not to build an overwhelming SAT solver, but rather to experiment a different *AOS* and validate *ExCoDyMAB* with an EA solving a general difficult combinatorial problem. Moreover, it is well-known that EAs do not perform well on SAT without using specialized operators. Nevertheless, the results of Table IV show that a basic EA using rather naive operators can indeed solve some instances. The main interesting result is that this set of benchmarks was difficult enough to highlight the benefit of using the proposed combination of *Compass* and *Ex-DMAB* rather than either separately – or a naive blind choice. The deliberate choice of several non specialized operators was also an important point to validate the control ability of *ExCoDyMAB* when facing variation operators of very different efficiencies. Competing for SAT race implies using highly specialized operators, and is left for further work.

VIII. CONCLUSION AND PERSPECTIVES

This paper has introduced *ExCoDyMAB*, an original method for doing *Adaptive Operator Selection* in Evolutionary Algorithms, i.e., to autonomously select between different variation operators according to the history of the search so far. This controller has the advantage of being able to deal with any operator, since it handles all parameters in an abstract way. It could be easily used, as well, with other search techniques that can use different 'move' or 'diversification' operators.

Two main features of *ExCoDyMAB* should be highlighted: the *Extreme Compass Credit Assignment*, that is obtained by the aggregation of three different performance measures (fitness, diversity, and execution time), considering the extreme values over a sliding window for the first two, and the mean value for the last one; and the *DMAB Operator Selection* rule, which effectively adapts to modifications in the operators preferences by means of a change detection test triggering a complete restart.

Both features come from different backgrounds and have proved to be very complementary. Using the Extreme values from the aggregated performance measure allows the algorithm to identify occasional but highly beneficial operators. The inclusion of population diversity besides the traditional fitness improvement measure contribute to escape from local optima.

This study highlights the importance of both control stages of *AOS*, namely credit assignment and operator selection rules. These two features both contribute to the overall performance of the proposed autonomous control method, explaining the efficiency gain compared to each previous method used alone.

One main drawback of *ExCoDyMAB* remains the tuning of its meta-parameters: even though it has been demonstrated here that off-line meta-parameter tuning using the F-Race paradigm can be done in a fraction (15%) of the time needed for a complete factorial DOE, off-line tuning remains quite costly. Furthermore, though the normalization of fitness improvements and diversity by *Compass* might result in a possible robust setting for the scaling parameter of the MAB balance (i.e., the value found here using Racing), further work is needed to get a deeper understanding of how to tune the meta-parameter γ that triggers the change detection test.

Finally, in order to assess the generality of the proposed approach, we will need to apply *ExCoDyMAB*, together with state-of-the-art variation operators, on a large series of SAT instances and to compare its results with those of the best available SAT solvers.

ACKNOWLEDGMENTS

This work was supported in part by the European STREP EvoTest (IST-33472) and by the Pays de la Loire regional project *RadaPop*.

REFERENCES

[1] F. Lobo, C. Lima, and Z. Michalewicz, Eds., *Parameter Setting in Evolutionary Algorithms*, ser. Studies in Computational Intelligence. Springer, 2007, vol. 54.

[2] L. Da Costa and M. Schoenauer, "GUIDE, a Graphical User Interface for Evolutionary Algorithms Design," in *GECCO Workshop on Open-Source Software (SoftGEC)*, J. H. Moore, Ed. ACM Press, 2007, software available at http://guide.gforge.inria.fr/.

[3] A. Eiben, Z. Michalewicz, M. Schoenauer, and J. Smith, *Parameter Setting in Evolutionary Algorithms*, ser. Studies in Computational Intelligence. Springer, 2007, vol. 54, ch. Parameter Control in Evolutionary Algorithms, pp. 19–46.

[4] V. Maniezzo, R. Battiti, and J.-P. Watson, Eds., *Learning and Intelligent Optimization*, ser. Foundations of Computing. Springer Verlag, 2008.

[5] E. K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, *Handbook of Meta-heuristics*. Kluwer, 2003, ch. Hyper-heuristics: An Emerging Direction in Modern Search Technology, pp. 457–474.

[6] J. Maturana and F. Saubion, "A compass to guide genetic algorithms." in *Proc. PPSN'08*, G. Rudolph et al., Ed. Springer, 2008, pp. 256–265.

[7] D. Goldberg, "Probability Matching, the Magnitude of Reinforcement, and Classifier System Bidding," *Machine Learning*, vol. 5, no. 4, pp. 407–426, 1990.

[8] A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag, "Extreme value based adaptive operator selection," in *Proc. PPSN'08*, G. Rudolph et al., Ed. Springer, 2008, pp. 175–184.

[9] L. Da Costa, A. Fialho, M. Schoenauer, and M. Sebag, "Adaptive operator selection with dynamic multi-armed bandits," in *Proc. GECCO'08*, M. Keijzer et al., Ed. ACM Press, 2008, pp. 913–920.

[10] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proc. ICGA'89*, J. D. Schaffer, Ed. Morgan Kaufmann, 1989, pp. 61–69.

[11] F. Lobo and D. Goldberg, "Decision making in a hybrid genetic algorithm," in *Proc. ICEC'97*, T. Bäck et al., Ed. IEEE Press, 1997, pp. 121–125.

[12] A. Tuson and P. Ross, "Adapting operator settings in genetic algorithms," *Evolutionary Computation*, vol. 6, no. 2, pp. 161–184, 1998.

[13] H. J. C. Barbosa and A. M. Sá, "On adaptive operator probabilities in real coded genetic algorithms," in *Proc. XX Intl. Conf. of the Chilean Computer Science Society*, 2000.

[14] B. A. Julstrom, "What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm on genetic algorithms," in *Proc. ICGA'95*, L. J. Eshelman, Ed. Morgan Kaufmann, 1995, pp. 81–87.

[15] J. M. Whitacre, T. Q. Pham, and R. A. Sarker, "Use of statistical outlier detection method in adaptive evolutionary algorithms," in *Proc. GECCO'06*, M. Cattolico, Ed. ACM, 2006, pp. 1345–1352.

[16] D. Thierens, "An adaptive pursuit strategy for allocating operator probabilities," in *Proc. GECCO'05*, H.-G. Beyer, Ed. ACM Press, 2005, pp. 1539–1546.

[17] Wong, Lee, Leung, and Ho, "A novel approach in parameter adaptation and diversity maintenance for GAs," *Soft Computing*, vol. 7, no. 8, pp. 506–515, 2003.

[18] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2/3, pp. 235–256, 2002.

[19] 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.

[20] E. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, pp. 100–115, 1954.

[21] 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 *Proc. LION'09*, 2009.

[22] S. A. Cook, "The complexity of theorem-proving procedures," in *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*. New York, USA: ACM, 1971, pp. 151–158.

[23] H. Hoos and T. Stützle, *SATLIB: An Online Resource for Research on SAT*. www.satlib.org: IOS Press, 2000, pp. 283–292.

[24] M. Birattari, "The problem of tuning metaheuristics as seen from a machine learning perspective," Ph.D. dissertation, Université Libre de Bruxelles, Brussels, Belgium, 2004.

[25] B. Yuan and M. Gallagher, "Statistical racing techniques for improved empirical evaluation of evolutionary algorithms," in *Proc. PPSN'04*, Xin Yao et al., Ed. Springer Verlag, 2004, pp. 172–181.