UNIVERSITÉ PARIS-SUD 11

CENTRE DE RECHERCHE COMMUN
INRIA MICROSOFT RESEARCH

Adaptive Operator Selection for Optimization

Ph.D. Thesis

École Doctorale d'Informatique, ED 427

Université Paris-Sud 11

By **Álvaro Roberto SILVESTRE FIALHO**

Microsoft Research – INRIA Joint Centre
28 rue Jean Rostand 91893 Orsay Cedex, France

Abstract

Evolutionary Algorithms (EAs) are stochastic optimization algorithms which have already shown their efficiency on many different application domains. This flexibility is achieved mainly due to the many parameters that can be defined by the user according to the problem at hand. However, EAs are very sensitive to the definition of these parameters, and there are no general guidelines for an efficient setting; as a consequence, EAs are rarely used by researchers from other domains. The methods proposed in this thesis contribute into alleviating the user from the need of defining two very sensitive and problem-dependent choices: which variation operators should be used for the generation of new solutions, and at which rate each operator should be applied. The paradigm, referred to as Adaptive Operator Selection (AOS), provides the on-line autonomous control of the operator that should be applied at each instant of the search, *i.e.*, while solving the problem. In order to do so, one needs to define a Credit Assignment scheme, which rewards the operators based on the impact of their recent applications on the current search process, and an Operator Selection mechanism, that decides which should be the next operator to be applied, based on the empirical quality estimates built by the rewards received. In this work, the Operator Selection problem has been tackled as yet another instance of the Exploration versus Exploitation dilemma: the best operator needs to be exploited as much as possible, while the others should also be minimally explored from time to time, as one of them might become the best in a further moment of the search; different Operator Selection techniques based have been proposed to extend the Multi-Armed Bandit paradigm to the very dynamic context of AOS. On the Credit Assignment side, rewarding schemes based on extreme values and on ranks have been proposed, in order to provide more robust operator assessments, while promoting the use of outlier operators. The different AOS methods formed by the combinations of the proposed Operator Selection and Credit Assignment mechanisms have been validated on a very diverse set of benchmark problems. Based on empirical evidences gathered from this empirical analysis, the final recommended method, which uses the Rank-based Multi-Armed Bandit Operator Selection and the Area-Under-Curve Credit Assignment schemes, has been shown to achieve state-of-the-art performance while also being very robust with respect to different problems.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AbsIns Absolute Instantaneous

AbsAvg Absolute Average

AbsExt Absolute Extreme

AOS Adaptive Operator Selection

AP Adaptive Pursuit

AUC Area-Under-Curve

BBOB Black-Box Optimization Benchmarking

ExCoDyMAB Extreme Compass - DMAB

DE Differential Evolution

DMAB Dynamic Multi-Armed Bandit

EA Evolutionary Algorithm

EC Evolutionary Computation

ECDF Empirical Cumulative Distribution Function

EP Evolutionary Programming

ERT Expected Running Time

ES Evolution Strategy

Ex-DMAB Ex-DMAB

EvE Exploration versus Exploitation

FAUC Fitness-based Area-Under-Curve

FSR Fitness-based Sum-of-Ranks

GA Genetic Algorithm

GP Genetic Programming

MAB Multi-Armed Bandit

NDCG Normalized Discounted Cumulative Gain

NormIns Normalized Instantaneous

NormAvg Normalized Average

NormExt Normalized Extreme

PH Page-Hinkley

PM Probability Matching

RMAB Rank-based Multi-Armed Bandit

ROC Receiver Operating Characteristic

SAT Boolean Satisfiability

SLMAB Sliding Multi-Armed Bandit

SR Sum-of-Ranks

TCR Total Cumulated Reward

TSP Traveling Salesman Problem

$\mathcal{TV}$ Two-Values

UCB Upper Confidence Bound

Part I

General Introduction

Chapter 1

Introduction

1.1 Context/Motivation

EAs are stochastic optimization algorithms remotely inspired by the Darwinian "survival of the fittest" paradigm. Let the goal be to optimize some objective function, referred to as *fitness* function, defined on search space X; elements of X are called *individuals*, and a set of individuals is termed a *population*. EAs evolve a population of individuals by iteratively (i) selecting some individuals (the *parents*), favoring those with better fitness; (ii) applying stochastic perturbations on the parents using some *variation operators*, thus generating *offspring*; (iii) evaluating the offspring (*i.e.*, computing their fitness values); and finally, (iv) selecting some individuals among the parents and the offspring to become the next parents, again favoring fitter individuals. This cycle is iterated until a satisfactory solution is found, or another stopping condition is attained. A more comprehensive description is presented in Chapter 2.

EAs consistently perform well approximating solutions to many different types of problems beyond the reach of standard methods (see, *e.g.*, all applications described in [T. Yu et al., 2008]; a track[1] on "Real-World Applications" is also yearly held within one of the main conferences of the field, the Genetic and Evolutionary Computation Conference), specially because they do not make any strong assumption about the problem to be solved, being able to handle structured and mixed search spaces, irregular, noisy, rugged, or highly constrained objective functions, etc. But, although demonstrating to be an exciting research field with the power to assist scientists, researchers and engineers in the task of solving hard optimization problems, EAs are rarely used outside the circle of *knowledgeable practitioners*; they still miss reaching the status of off-the-shelf tools. There are several reasons for that, all boiling down to *a lack of practical support* when it comes to actually design an EA for a given application. On a conceptual level, despite Michalewicz' seminal book [Michalewicz, 1996] and the two more recent books by [Eiben and Smith, 2003] and [DeJong, 2006], the terminology used by authors still reflect the evolutionary trend they historically belong to. On a practical level, while some software packages provide a unifying framework for the various evolutionary approaches

[1] RWA track on GECCO'10: `http://sigevo.org/gecco-2010/organizers-tracks.html#rwa`

(see, *e.g.*, the EO [Keijzer *et al.*, 2002] and the GUIDE [Collet and Schoenauer, 2003; Da Costa and Schoenauer, 2009] initiatives), the success of EAs is still very sensitive to the setting of quite a few parameters, *e.g.*, main and offspring population sizes, types of variation operators and respective application rates, types of selection mechanisms and related parameters.

In early days, Evolutionary Computation (EC) actually benefited from those numerous parameters, which ensure their outstanding flexibility, and make them applicable to the mentioned wide spectrum of applications. The contemporary view of EAs, however, acknowledges that specific problems require specific setups for satisfactory performance [Eiben *et al.*, 2007]: when it comes to solving a given problem, parameter setting is viewed as the Achilles' heel of EAs, on par with their high computational cost. From these observations, a current trend in EC is to focus on the definition of more autonomous solving processes, which aim at enabling the basic user to benefit from a more efficient and easy-to-use algorithmic framework. Parameter setting in EAs appears thus as a major issue that has deserved much attention during recent years [Eiben *et al.*, 1999; Eiben *et al.*, 2007], and research is still very active nowadays, as witnessed by a complete edited book that has been recently published [Lobo *et al.*, 2007], and by the numerous recent references cited in this document. Interestingly, the search for algorithmic technologies enabling the (naive) end-user to benefit from good performances through autonomous parameter setting is also considered as a priority in neighbor fields such as operation research or constraint programming [Hutter *et al.*, 2006; T. Stützle et al., 2009]; in the same way than in EC, these fields involve sophisticated solver platforms, requiring an extensive expertise in order to be used to their fullest extent. Chapter 3 presents a summary about the current state of research in parameter setting of EAs.

Some of the user choices that most affect the performance of EAs concern the variation operators: which operators should be used for the generation of new solutions, and at which rate each of the chosen operators should be applied. These choices affect the way in which the algorithm will explore the search space while also being able to efficiently exploit the most promising regions, the so-called Exploration versus Exploitation (EvE) dilemma. Such definition is usually done by following the user's intuition, or by using an off-line tuning procedure aimed at identifying the best operator for the problem at hand. Besides being computationally expensive, off-line tuning however generally delivers suboptimal performances. Intuitively, the EA should proceed from a global (early) exploration of the landscape to a more focused exploitation-like behavior, as already empirically and theoretically demonstrated (see, *e.g.*, [Eiben *et al.*, 2007] and references therein). Thus, its parameters values should be varied accordingly, while solving the problem: more exploratory operators should be preferred in the earlier stages of the search, and more priority should be given to the fine-tuning/exploitation operators when approaching the optimum.

The on-line parameter setting in EAs is often called as Parameter Control [Eiben *et al.*, 2007]. This is the context in which the contributions presented in this thesis are inserted, more specifically on AOS, which can be summarized as follows.

1.2 Main Contributions

In essence, the goal of AOS is to select on the fly the best operator at each stage of the search, *i.e.*, the operator that is currently maximizing some measure of quality, usually, though not exclusively [Maturana and Saubion, 2008a; Maturana *et al.*, 2009a], reflecting the fitness improvement brought by its application. AOS thus raises two main issues, the *Operator Selection* and the *Credit Assignment*, which will be explained in the following, together with the contributions we propose in this thesis to address each of them.

1.2.1 Operator Selection

The first issue, the *Operator Selection*, defines how the next operator to be applied should be selected, based on its known empirical quality. Indeed, it might be seen as yet another level of the EvE dilemma. While an operator that has performed well in the recent past should certainly be used again (exploitation), other operators that did not perform so well should also be tried (exploration). The rationale for exploration is rooted, firstly, in the stochastic nature of the evolutionary process (some seemingly poorly performing operators might just have been unlucky); and secondly, on its dynamics: the quality of an operator depends on the region of the fitness landscape being explored by the current population, *i.e.*, good operators might become poor as evolution goes on, and vice-versa.

Notably, the EvE trade-off has been intensively studied in the context of Game Theory, in the so-called Multi-Armed Bandit (MAB) framework [Lai and Robbins, 1985]. The Upper Confidence Bound (UCB) [Auer *et al.*, 2002] is a MAB algorithm that provides asymptotic optimality guarantees with respect to the total cumulated reward in a stationary context. However, as previously mentioned, the AOS context is extremely dynamic. The main *contribution* of this thesis, in summary, lies in the proposal and analysis of schemes to solve the AOS problem based on the UCB algorithm; we have proposed different extensions to it, in order to enable it to efficiently cope with the dynamics of evolution and with the very different characteristics of the problems to be tackled. These extended MAB schemes are detailed in Chapter 5.

Starting from the original UCB algorithm (referred to as the original or standard MAB algorithm in the following, for the sake of convenience), presented in Section 5.3.1, our first proposal to extend it to the dynamic context of AOS was the Dynamic Multi-Armed Bandit (DMAB) algorithm [Da Costa *et al.*, 2008], which proceeds by coupling the original MAB technique with a statistical change-point test, the Page-Hinkley test [Hinkley, 1970]: upon the detection of a change in the operator quality distribution, the MAB process is restarted from scratch.

Although showing to be very efficient, the DMAB required the tuning of a very sensitive and problem-dependent hyper-parameter, the threshold value for the change-detection test. This led to the proposal of a smoother way to account for dynamic environments in the MAB framework, referred to as Sliding Multi-Armed Bandit (SLMAB) [Fialho *et al.*, 2010a], which uses a sliding time window to gracefully update the operator quality estimates, discarding ancient events while preserving the information from the recent events. Contrasting with DMAB, the SLMAB does not call upon an external

monitoring of the evolution process, involving only 2 hyper-parameters (DMAB has 3).

The latest proposal concerning the *Operator Selection* part is what we refer to as Rank-based Multi-Armed Bandit (RMAB), in which the evaluations provided by a rank-based *Credit Assignment* scheme (that is part of the contributions mentioned in the following sub-section) are used directly in the place of the UCB empirical estimation. In this way, as the rewarding of one operator affects the ranks, and consequently the quality assessments, of all the other operators, this technique is already dynamic by definition, while being very robust with respect to its hyper-parameters, as better discussed in the following.

1.2.2 Credit Assignment

All the previously mentioned bandit-based *Operator Selection* methods (and other existent approaches for the same purpose) select the operator to be applied next based on some assessment of their respective qualities. Defining how to estimate their quality based on the impact brought by their most recent applications is what we refer to as *Credit Assignment*, the second issue to constitute an AOS algorithm, which was also object of analysis and development in this thesis.

The most common way of assigning credits is to account for the fitness improvements brought by the operators applications. The use of the *instantaneous* value, *i.e.*, the latest fitness improvement achieved by the operator, is known to be an unstable measure, due to the stochastic nature of operators (one operator might have been unlucky on its latest trial, although being a very good option in the longer term). To alleviate such effect, the *average* of the latest rewards is more commonly used; however, by using such kind of assessment, operators that regularly achieve very small improvements are preferred instead of the operators that get rare but highly beneficial improvements. Motivated by other complex systems (*e.g.*, rogue waves, financial market, etc), and also by another empirical analysis within EAs [Whitacre *et al.*, 2006], in this thesis we support that the latter operator should be preferred in the place of the former, what can be achieved by the use of Extreme values, *i.e.*, the maximum reward recently received by the operator. Better results have been achieved when compared to the usual Instantaneous and Average schemes.

Nevertheless, with the use of the raw values of the fitness improvements, the AOS schemes implementing these *Credit Assignment* mechanisms need to have their behavior, which is controlled by a couple of hyper-parameters, tuned for each new problem. This is partially solved with the use of a simple normalization scheme; but even when normalized, the rewards are still based on the fitness values to some extent. In order to have a controller robust to many different situations, two *Credit Assignment* schemes based on ranks were proposed [Fialho *et al.*, 2010c; Fialho *et al.*, 2010b], namely, the Area-Under-Curve (AUC) and the Sum-of-Ranks (SR). Besides being rank-based, both of them, when considering the fitness values instead of the fitness improvements for the ranking, are completely comparison-based, *i.e.*, invariant with respect to monotonous transformations over the original fitness function, thus maintaining this very important property, which is already guaranteed by construction in most of the recent EAs (see, *e.g.*, an extensive mathematical analysis of the advantages of comparison-based randomized heuristics

presented in [Gelly *et al.*, 2007]; and the Covariance Matrix Adaptation - ES (CMA-ES) [Hansen and Ostermeier, 2001], a comparison-based state-of-the-art adaptive method for Evolution Strategies (ESs), that is also invariant with respect to rotations of the search space).

1.2.3 Empirical Validation

The different combinations of these proposals to *Operator Selection* and *Credit Assignment* gave origin to novel AOS methods. A last contribution of this thesis concerns their empirical validation. In order to do so, some artificial scenarios were proposed, which enable a detailed analysis of the behavior of each AOS method with respect to different situations.

The proposed AOS combinations have been compared between each other and with other baseline techniques on many different scenarios, as presented in Chapter 6: on the proposed artificially generated scenarios (Section 6.3), on some boolean benchmark problems (Section 6.4), on a comprehensive set of single-objective continuous problems (Section 6.6), and on a set of Boolean Satisfiability (SAT) instances (Section 6.5). In the latter case, an aggregation of fitness and diversity, named Compass, was used for as *Credit Assignment* [Maturana *et al.*, 2009a; Maturana *et al.*, 2010a], considering the fact that in multi-modal problems some diversity should always be maintained in order to avoid premature convergence.

The Rank-based Multi-Armed Bandit techniques have shown to be very efficient, while being also very robust to the many different situations in which they were assessed. After all the empirical evidences gathered, they are thus the recommended choices, by the time of the preparation of this manuscript, if one wants to implement AOS within a given algorithm/problem.

It is also important to note that the AOS paradigm is not exclusive to the EC framework. Indeed, any stochastic/local search algorithm that has different options for the exploration of the search space might profit from the proposed methods. Besides, the same paradigm can be directly used in the hyper-heuristics level, *i.e.*, selecting between different heuristics, instead of selecting between different variations of a given heuristic. These and more conclusions and ideas for further work are presented in Chapter 7, which concludes this manuscript.

1.3 Organization

The remainder of this thesis manuscript is organized as follows. In the first part, a review of the context of this work will be done, starting with a more comprehensive presentation of EAs in Chapter 2, with more focus being given to the EA variants used in the experimental section, namely, Genetic Algorithms and Differential Evolution. Then, a summary of all the background concerning the research area of parameter setting in EAs and other meta-heuristics will be presented in Chapter 3. In Chapter 4, a more detailed description and bibliographic review of the parameter setting sub-problem to which the contributions presented in this thesis are devoted to, the Adaptive Operator Selection, will be presented.

In the second part, the contributions of this thesis work will be presented. Chapter 5 will present and describe in detail the proposed AOS techniques, and Chapter 6 will describe and analyze the empirical evaluation done on some very diverse scenarios. Finally, Chapter 7 will conclude this thesis, summarizing the contributions, and pointing out possible directions for further work.

Part II

Background Review

Chapter 2

Evolutionary Algorithms

Contents

In this Chapter, we present an overview of EAs, depicting their general behavior and parameters. Besides, some popular EA variants are described, and some examples of application domains are presented.

2.1 Introduction

An important source of inspiration for the development of computational methods to automate problem solving, nowadays, is the "intelligent" way in which biological processes solve complex problems found in nature. Some popular examples of these approaches, referred to as bio-inspired methods, are: Neural Networks [Arbib, 2002], based on the structure of the biological brain; Fuzzy Logic [Klir and Yuan, 1995], inspired on the human way of reasoning; Swarm Intelligence algorithms, supported on living examples of collective social behavior, *e.g.*, the Particle Swarm Optimization (PSO) [Eberhart *et al.*, 2001] and the Ant Colony Optimization (ACO) [Dorigo *et al.*, 1996] methods; and finally, the Evolutionary Algorithms (EAs), which are global optimization methods that mimic the Darwinian "survival of the fittest" paradigm in order to solve optimization and search problems.

Since the seminal works trying to apply the evolution theory to optimization (we refer the reader to the edited book [Fogel, 1998] for a compilation of them), many variants of EAs have been independently developed around the world, originally for different domains of application, with the main difference being the representation and the variation operators used. Historically speaking, the pioneer methods were: Genetic Algorithms (GAs) [Holland, 1975; Goldberg, 1989], Evolution Strategies (ESs) [Rechenberg, 1972; Schwefel, 1981], Evolutionary Programming (EP) [Fogel *et al.*, 1966; Fogel, 1995], and more recently, Genetic Programming (GP) [Koza, 1992; Koza, 1994]. Research has been very active on GAs, ES and GP, but EP has been gradually disappearing from the literature, as it can be considered as a special case of ES. Besides, other popular techniques have been more recently created within the EA community, such as the Differential Evolution (DE) [Storn and Price, 1997; Price *et al.*, 2005], and the already mentioned PSO [Eberhart *et al.*, 2001] and ACO [Dorigo *et al.*, 1996] methods. Except for the latter two, which do not exactly follow the evolution paradigm, each of the mentioned techniques will be separately described in Section 2.4, with a more extensive presentation being done for the methods used in the experimental section of this manuscript (see Chapter 6), namely GAs and DE. The contemporary view, however, is that, given the continuous development and frequent hybridizations between the ideas proposed by each historical technique, it is becoming more and more difficult to differ between them, what justifies the recent proposal of a "unified view" for the different EA methods [DeJong, 2006]. Indeed, they all share the same *modus operandi*, as described in Section 2.2.

EAs have already demonstrated their efficiency on a very wide range of optimization problems (more on this in Section 2.5) beyond the reach of standard methods, *e.g.*, involving structured and mixed search spaces, irregular, noisy, rugged or highly constrained fitness functions, etc. This flexibility is mainly due to the several parameters, described

in Section 2.3, that can be tuned by the user according to the problem at hand. However, these same parameters are also the main responsible for the fact that EAs are rarely used by scientists from other domains, as there are no standard methods or guidelines for their setting. This is why research on automatic parameter setting methods is very active nowadays, as reviewed in Chapter 3. In Chapter 5 of this thesis, we present some contributions to one of the parameter setting sub-problems, referred to as Adaptive Operator Selection (AOS), surveyed in Chapter 4.

2.2 Modus Operandi

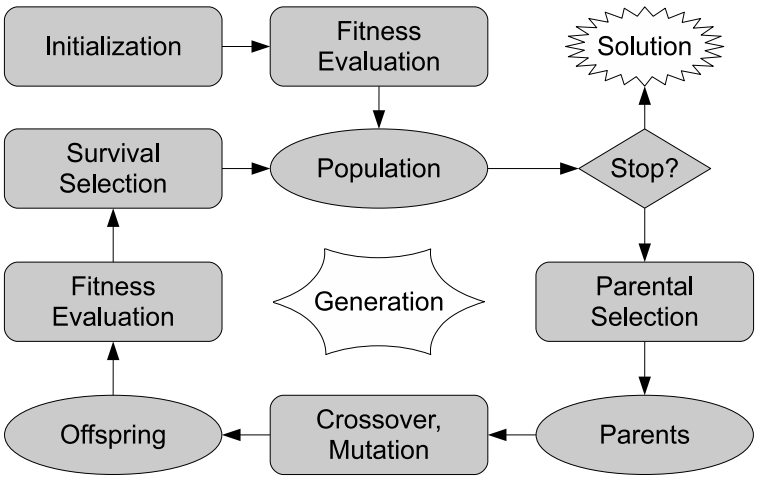


Figure 2.1: General cycle of Evolutionary Algorithms (EAs).

The different variants of EAs follow the same general outlines, depicted in Fig. 2.1, differing only in a few technical details, as detailed in Section 2.4. Generally speaking, thus, the *modus operandi* of the EAs can be described as follows.

1. A set (*population*) of candidate solutions (*individuals*) is initialized, usually representing a random sampling of the search space.

2. Each individual is then evaluated, according to the *fitness* function, which defines the problem objective: higher is the degree of achievement of a given candidate solution with respect to the problem at hand, "fitter" it is.

3. If none of the stopping criteria are satisfied (*e.g.*, optimal solution found, or total computational budget spent), go on to the next steps.

4. The first Darwinian natural *selection*-based process takes place. Individuals are selected as *parents* to reproduce, usually based on the fitness evaluation, as in nature: stronger/fitter individuals, *i.e.*, better candidate solutions, have higher chances of being selected for reproduction.

5. These selected individuals are then subject to blind variations (blind in the sense that no information about the problem or the consequences of the variation are considered), by the application of stochastic operators, namely *crossover* (recombination) and *mutation* operators, generating *offspring*.

6. The newly generated offspring are then evaluated, according to the same fitness function, which defines the problem.

7. Then comes the second and final Darwinian process, the replacement or *survival selection*, that defines which individuals, from both the parental population and the newly generated offspring population, will survive to the next iteration (*generation*) of the algorithm.

8. From this evolved population, a new generation can start, going back to Step 3.

From this general cycle, it can be seen that the evolution itself happens mainly due to two opposite forces, as follows. On the one hand, there is the application of *mutation* and *recombination operators*, which introduce random variation in the population, consequently performing an exploration of the solutions search space; intuitively, their sole application would lead to a random search. On the other hand, as in the inspiring theory, better candidate solutions (*i.e.*, fitter individuals), have higher chances to be used in the generation of new (hopefully also fitter) solutions, and to survive for the next generations; these Darwinian procedures are the responsible for giving a direction to the originally random search, leading it to the most promising regions of the search space. This process of *blind variation + natural selection* is then iterated until an optimal solution arises, or another stopping criterion is attained.

Besides being bio-inspired, EAs are thus stochastic algorithms that work by following a kind of *generate-and-test* (also known as *trial-and-error*) approach [Eiben and Schoenauer, 2002], as many other meta-heuristics, *e.g.*, Simulated Annealing [P.J.M. Laarhoven et al., 1987]. While describing the general cycle, several structures and procedures were mentioned; they are described into more detail in the following.

2.3 Components

EAs have mainly three kinds of components: some are related to the problem to be solved (Section 2.3.1), others depend on the representation being used (Section 2.3.2), while others are totally general (Section 2.3.3). Each of these groups will now be briefly described in turn.

2.3.1 Problem-dependent Components

The components that need to be defined according to the optimization problem can be described as follows.

Evaluation/Fitness Function

The evaluation or fitness function plays the role of the environment in the Darwinian natural selection-like procedures, assigning a score to each individual according to its degree of "achievement" with respect to the optimization problem at hand. The fitness function is thus the core of the algorithm, which needs to be very carefully designed, as it is often the only source of information about the problem that is available to the algorithm [Eiben and Schoenauer, 2002].

Although EAs are said to be robust with respect to very different situations (*e.g.*, irregular, noisy, and highly rugged fitness landscapes), a minimal level of continuity and/or regularity (the search gradient) needs to be provided to guide the search towards the most promising regions of the search space; otherwise it tends to act as a random search, with no direction to follow. In some cases, however, the definition of the fitness function itself is a very complex task; a recent paradigm, referred to as Interactive Evolutionary Computation (IEC), address this issue by "outsourcing" the fitness evaluation to humans, as in [Quiroz *et al.*, 2007], for example.

The fitness evaluation of a candidate solution is undoubtedly the most computationally expensive step of the EA cycle, and its computational cost affects other user choices, mainly the size of the population and the number of offspring created at each generation, as each offspring requires a fitness evaluation. In case the fitness evaluation cost becomes prohibitive for evolution to take place (usually many generations, consequently fitness evaluations, are needed), some approximations between the fitness values found in the neighborhood of the candidate solution under assessment might also be used, as in [Martikainen and Ovaska, 2006]. Besides, in some application fields, the evaluation might also be very noisy, thus requiring the averaging of several independent assessments in order to have a reliable measure of quality.

In cases where there is more than one objective to be optimized, referred to as *multi-objective* in the literature, special fitness assessments need to be used to take into account all the objectives; the most popular criterium is the *Pareto optimality* (see, *e.g.*, [Deb, 2001; Mueller-Gritschneder *et al.*, 2009]).

Representation

From the structural point of view, in order to solve a given problem, the main issue that needs to be defined is how the candidate solutions are going to be represented. The solutions themselves, referred to as *phenotypes*, might be very complex structures; but their corresponding low-level representation, the *genotypes*, which are the structures manipulated by the algorithm, are usually much simpler. As in the inspiring theory, genotypes are constituted by *genes*, which store the values of the candidate solution for each variable of the problem at hand. The most common representation or *encoding* schemes can be listed as follows.

- Binary encoding: Vectors of binary values, or bit-strings, are commonly employed to represent problem solutions that have just two possible values for each variable. For example, in the combinatorial SAT problem [Cook, 1971], which consists in assigning

values to binary variables in order to satisfy a Boolean formula, each gene represents the boolean state of each variable of the problem [Lardeux *et al.*, 2006].

- Permutation encoding: Vectors of integers are usually used to ordering problems. The classical application example is the well-known Traveling Salesman Problem (TSP) problem, in which there is a set of cities that need to be visited by a salesman, the objective being to find the order of cities that minimize the distance to be travelled. In this case, each city is assigned an integer number, and the order of these numbers defines the order in which the salesman will visit the cities [Merz and Freisleben, 1997].

- Real-value encoding: For some problems, the direct use of real values is preferred, as for example, to optimize the weights of a neural network, with each gene of a candidate solution representing the value of each of the corresponding weights [Obradovic and Srikumar, 2000].

- Tree encoding: It is used mainly to evolve programs or regular expressions, with every solution being encoded as a tree of objects such as functions or commands of a given programming language. For example, given a set of input and output data samples, it can be used to find a function that maximizes the mapping between them [Koza *et al.*, 2003].

The permutation and the binary encoding schemes are historically used by GAs to solve combinatorial problems, as confirmed by the given examples. The real-value encoding is usually employed by ES and DE on continuous optimization problems; while the tree representation scheme is often used within GP to automatically generate or optimize programs. More details about each of these mentioned EAs will be given in Section 2.4.

2.3.2 Representation-specific Components

Some of the components of an EA, namely the initialization procedure and the variation operators, are representation-dependent, *i.e.*, they need to be defined according to the chosen representation model. This is in fact one of the reasons why EAs are successfully applied to so many different domains of application (see Section 2.5 for a few examples): given an appropriate initialization procedure and variation operators, any kind of search space can be tackled [Eiben and Schoenauer, 2002]. Such representation-specific components will be briefly described in the following.

Initialization

According to the representation being used, the initial population is usually created after a random sampling of the search space. A uniform sampling is commonly used when the search space is finite and its bounds are known, *e.g.*, in the binary, permutation and tree-based representations. For the real-value representation, the uniform sampling can also be used if the search space bounds are provided; a gaussian distribution being used for the initialization otherwise.

Furthermore, in case some prior knowledge is available, it might be used in the initialization process, *e.g.*, by directly including a known good solution. But, on the one hand, such manipulated initialization might result into a wrong bias to the search process, what is of course much worst than having no bias at all [Eiben and Schoenauer, 2002]; and, on the other hand, this extra effort is usually not very well paid-off as, when starting from a random population, the same EA would typically need just very few generations to achieve the same level of solution quality [Eiben and Smith, 2003].

Variation Operators

Also representation-dependent, there are different kinds of variation operators, as follows.

Mutation operators are the *asexual* variation operators, *i.e.*, a single parent individual is considered to generate an offspring. These operators are the responsible for introducing non-existing (or re-introducing missing) characteristics into the population, thus augmenting the so-called *genetic diversity*. A complementary view for their purpose is that of *fine-tuning*: individuals might improve their respective qualities after suffering slight variations (*e.g.*, mutation of a single gene). Traditional mutation operators for each of the four popular representations mentioned in Section 2.3.1 can be listed as follows.

- For bit-strings, the bit-flip mutation operator flips each bit with probability $1/\ell$ by default (although a different probability can be employed), ℓ being the length of the bit-string; other popular mutation operator is the x-bit, which flips x randomly chosen bits each time it is applied.
- For real-valued vectors, the most common mutation operator is the addition of a random value to each vector component or gene. It is mainly used within ES, with the random value being extracted from a normal distribution with zero mean and a pre-defined standard deviation (also referred to as the mutation step-size). In the case of DE, several mutation strategies exist, using the differences between two or more vectors (individuals) in different ways for perturbing the vector population.
- For permutation-like and trees, a popular mutation operator is the order changing: two genes are randomly selected and have their values exchanged. In the case of trees, not just the values of the chosen nodes, but also both sub-trees (or branches) attached to them, are usually switched. A simpler alternative is the exchange of the value found in the chosen gene or node by another value randomly chosen from the finite search space.

Crossover or recombination operators are the *sexual* variation operators: parts of the genetic material of two or more different parent individuals are recombined somehow, creating one or more new offspring. Its use is justified by the *building blocks* assumption: supposedly, the good fitness scores of the parents are related to some portions of their genetic material; with strictly positive probability, the good portions (building blocks) of both parents are recombined, consequently creating a fitter individual. Accordingly, the most common crossover operator for all the representations is the x-point crossover, which

divides each parent individual into x building blocks, forming the offspring by different recombinations of these portions; a more exploratory variant is the uniform crossover, which uniformly selects which genes are taken from each parent to constitute the new offspring. In the same way than for the mutation operators, there are other ways of doing so as well, according to the representation being used; for example, in the case of real-valued vectors, arithmetical operations might be done between the genes of both parents; while in tree-like representations, different branches of the parents trees can be exchanged. It is important to note that the effect of the crossover operators on the search process is automatically adapted, by construction, according to how converged the population is: while there is a good level of diversity, it helps into exploring the search space; the less diversity there is, the more exploitation-like will be its behavior, up to the total inefficiency as, differently from the mutation counterparts, it can not introduce any novelty into the population.

The standard mutation and crossover operators are pure stochastic transformations that receive as input one or more (parent) individuals, and generate as output one or more new (offspring) individuals, not using any feedback about the impact of their application on the search; due to this, their application is usually referred to as a *blind* variation. Then, it is usually up to the replacement selection mechanism to accept or not the generated offspring, and consequently guide the search process. Differently from that, in some well-known application domains, the available information about the problem might be used into the design of specialized or "intelligent" operators. For example, the flipping of a bit might be prevented by the fact that it is known (by simulating the results of its flipping) to be already set to a good value, as done in the SAT domain (see, *e.g.*, the GASAT [Lardeux *et al.*, 2006]); or by intelligently choosing which building blocks should be exchanged, as in most of the specialized crossover operators for the TSP problem (see, *e.g.*, [Chan *et al.*, 2005]). By using trial-and-error and feedback from the search in order to decide its move, what is being done by these operators is in fact what is usually referred to as "local search". In other cases, some well-performing meta-heuristics are also used as inspiration for local search variation operators within EAs, as in [Branke *et al.*, 2003], which proposes the use of crossover operators based on the Ant Colony Optimization algorithm for the same TSP problem. EAs that employ local search techniques as variation operators are commonly called as Memetic Algorithms (MA) [Krasnogor, 2002].

2.3.3 General Components

Agreeing with the "unified view" of EAs proposed in [DeJong, 2006], some components are general, not being affected by the kind of representation used. These components can be listed as follows.

- the size of the parent population m;
- the size of the offspring population n;
- the procedure for selecting parents *pselect*;
- the procedure for producing offspring *reprod*;

- the procedure for selecting survivors *sselect*;
- the stopping criterion *cstop*.

Each of these representation-independent components will be briefly described in the following.

Parent and Offspring Population Sizes

The parent population size m defines the level of parallel search done by the EA, as it contains the starting points for the new solutions explored in the search space at each generation [DeJong, 2007], while the offspring population size n determines the number of trials done by the algorithm at each generation.

Their definition is mainly related to (i) how rugged the fitness landscape is thought to be, as a bigger population will enable a better parallel exploration of multiple peaks; and to (ii) the available computational budget, as at each generation it is required to evaluate the fitness of all newly generated individuals, which is by far the most expensive step of the evolutionary cycle, as previously discussed.

Parental or Reproduction Selection

One of the Darwinian representation-independent steps that "guide" the evolution engine, the *parental* selection *pselect*, as its name says, is the procedure responsible for selecting which of the individuals will be chosen for reproduction.

A very simple and popular parental selection method is the *proportional* one: for each individual, the probability of being selected is proportional to its fitness score, with a roulette wheel-like method over these probabilities being used for selection. However, in some application domains an individual might have a fitness value that is orders of magnitude higher than the others. By using the proportional method in such a case, this super-individual will very probably be always selected for reproduction, thus quickly taking over the entire population, consequently leading to (possibly) premature convergence.

To avoid such kind of problem related to fitness ranges, other methods widely used nowadays are: (i) the *rank-based* selection, in which the probability of being selected of an individual is proportional to its fitness ranking with respect to the other individuals in the population; (ii) the *tournament* selection, in which T individuals are uniformly chosen from the population, and the best between these T individuals is selected, with $T \in [2, m]$, m being the parent population size; and a different variant, the (iii) *stochastic tournament* selection, in which 2 individuals are randomly chosen, and the best between them is selected with probability $t \in [0.5, 1]$.

Survival or Replacement Selection

The other representation-independent procedure that enforces the simulation of the Darwinian natural selection process is the *survival* selection, *sselect*, which defines how the population of the next generation will be constituted, based on the current parental and

offspring populations, *i.e.*, which individuals of both populations are going to survive for the next generation.

Broadly speaking, there are two categories of replacement methods: (i) individuals from both populations are considered, thus disputing between each other for survival, as the number of available "places" in the next generation is limited (the main population size m); or (ii) just the offspring population is considered, and the best m out of n individuals are maintained for the next generation. The former is referred to as the "plus" strategy in the Evolution Strategies (ESs) context, and is elitist by default, as the best individuals out of both populations are maintained. The latter is called as the "comma" strategy; if the risk of possibly losing the best solution found can not be assumed, elitism should be externally added in this case (remembering that the maintenance of the best in the population will always create a bias in that direction, what might be good, or not).

In case both parent and offspring populations have the same size ($m = n$) and a *comma*-like survival selection is used, the EA is said to be *generational*, *i.e.*, the entire population is replaced after each generation. When $m > n$ and $n = 1$, with a *plus*-like survival selection, the algorithm is referred to as being *steady-state*, behaving in a much greedier way.

Termination Condition

The termination condition for EAs is commonly related to the available budget, *e.g.*, elapsed time, number of generations or fitness evaluations. However, part of this budget is usually wasted somehow, as follows. As soon as the population *converges*, *i.e.*, most of the individuals are very similar (no much diversity can be found into the population), the search becomes very inefficient, with the best solution being improved just by a lucky move (random sampling = Monte Carlo).

A more intelligent stopping criterion relies on the convergence measure: as soon as it is detected, the search can be stopped, possibly restarting from a new random initial population in order to provide more opportunities for the algorithm to find a better solution within the same available computational budget. There are several ways to account for population convergence, the simplest one is the number of generations since the last time a better solution was found (stagnation). A more complex mechanism that can be found in the GA literature is the "bit-wise average convergence measure" [Goldberg *et al.*, 1995], which estimates, for each gene, the percentage of individuals presenting the same value, with the final measure being the average of the values found on all genes; convergence is detected when this average exceeds some user-defined threshold.

General Representation of Special Cases

After the definition of these general/representation-independent components, it becomes possible to easily describe EAs well-known to the community, and also to create new variations [DeJong, 2007]. Besides facilitating the human comprehension, this general description is also beneficial in practice, when implemented into existing toolkits, as in the *Evolving Objects* (EO) library [Keijzer *et al.*, 2002], or in the more recent *GUIDE*

[Da Costa and Schoenauer, 2009].

For example, a tentative of canonical GA could be described as follows:

- $m = n$;
- $pselect$ = probabilistic, fitness-proportional (although tournament is more popular nowadays);
- $reprod$ = crossover and mutation;
- $sselect$ = deterministic, offspring only (equivalent to the "comma" or "generational" replacement).

Another well-known EA, the standard $(\mu + \lambda) - ES$, corresponds to:

- $m = \mu$;
- $n = \lambda$;
- $pselect$ = uniform;
- $reprod$ = mutation;
- $sselect$ = deterministic truncation ("plus" replacement method).

2.4 Popular EA Variants

Several EA variants exist, the most popular ones will be briefly described in the following, namely, Evolution Strategies (ESs), Evolutionary Programming (EP), Genetic Programming (GP), Genetic Algorithms (GAs) and Differential Evolution (DE). A more detailed description will be done for the latter two, as they are the EAs used in the experimental section of this manuscript (Chapter 6); although the Adaptive Operator Selection techniques proposed in this thesis, presented in Chapter 5, could be straightforwardly extended to any of the other EA methods. For the sake of correctness, the "historical" view of the techniques will be used in their description; however, as already discussed, given the frequent hybridizations and exchanges between the areas, nowadays it is becoming more and more difficult to differ between them, a "unified view" of EAs being recommended [DeJong, 2006].

2.4.1 Evolution Strategies

Evolution Strategies (ESs) [Rechenberg, 1972; Schwefel, 1981] are very popular EAs; they are mostly applied to numerical (continuous) optimization problems, thus using real-valued vectors to represent the solutions.

The common alternatives for the crossover are the exchange or the linear recombination of components, although historically no crossover is used. Oppositely, the mutation is always applied, being usually a gaussian noise with zero mean and user-defined standard

deviation (also referred to as mutation step-size). Both *comma* and *plus* selection strategies are considered, as well as different sizes for the parental and offspring populations, according to the characteristics of the problem.

It is worthy noting that the state-of-the-art continuous optimizer to date is the Covariance Matrix Adaptation - ES (CMA-ES) [Hansen and Ostermeier, 2001], an ES with a very efficient dynamic control of the mutation step-size, shape and direction. The CMA and other schemes for automatically adapting the mutation step-size will be briefly discussed in Section 3.3.4.

2.4.2 Evolutionary Programming

Evolutionary Programming (EP) was originally applied to the evolution of finite state automata for machine learning problems [Fogel *et al.*, 1966], with representation and variation operators being specially designed to this kind of search space. More recently, however, it was adapted to also tackle numerical optimization problems [Fogel, 1995], with just few details differing it from the ESs, *e.g.*, stochastic instead of deterministic replacement. Given the higher popularity of ESs, EP is rarely mentioned in the recent literature; most authors consider it nowadays as a special case of ES.

2.4.3 Genetic Programming

Genetic Programming (GP) [Koza, 1992; Koza, 1994] is known as the EA variant to be used when evolving programs and logical expressions, using trees with varying sizes to represent them. Some Lisp-like languages that naturally embody tree structures are frequently used within GP; although other functional languages can also be adapted somehow to do so.

The evolution engine is very similar to GAs, which will be presented in Section 2.4.4; special crossover and mutation operators are employed, in order to be able to cope with the tree representation. As the individuals do not have a fixed size, a very common problem in GP is the so called *bloat*, the uncontrolled growth of an individual, with a comprehensive body of literature being dedicated to its control (see, *e.g.*, [Luke and Panait, 2006]).

2.4.4 Genetic Algorithms

Genetic Algorithms (GAs) [Holland, 1975; Goldberg, 1989] are by far the most popular methods. Traditionally, they are used to address combinatorial problems, using the very general bit-string representation. Other representations can also be employed to facilitate the translation from phenotype to genotype, consequently making easier the manipulation of the candidate solutions. For example, for ordering problems like the Traveling Salesman Problem (TSP), the permutation-based representation is used, in which each gene i corresponds to the object considered as being in the ith position. A very general representation of a GA in the form of a pseudo-algorithm is shown in Algorithm 2.1.

Each new offspring is usually generated as follows. Firstly, the parents are selected according to the parental selection method used, *e.g.*, the tournament selection method. A crossover operator is then applied with probability p_c over the two selected parent

Algorithm 2.1: General pseudo-algorithm for a Genetic Algorithm

```
1: Generate the initial population
2: Evaluate the fitness of all individuals
3: while the stopping condition is not satisfied do
4:    for i = 1 to n do
5:       parent1 = ParentalSelection(parentPop)
6:       if rndreal[0, 1) < p_c then
7:          repeat
8:             parent2 = ParentalSelection(parentPop)
9:          until parent2 != parent1
10:         offspringPop[i] = Crossover(parent1, parent2)
11:      else
12:         offspringPop[i] = Copy(parent1)
13:      end if
14:      if rndreal[0, 1) < p_m then
15:         offspringPop[i] = Mutation(offspringPop[i])
16:      end if
17:   end for
18:   Evaluate the fitness of all the generated offspring
19:   parentPop = SurvivalSelection(parentPop, offspringPop)
20: end while
```

individuals; the most common crossover operators in this case are the x-point and the uniform ones. The resulting offspring (a copy of one of the parents in case crossover is not applied) is then subject to a mutation operator, *e.g.*, a bit-flip or a x-bit mutation, with probability p_m. Finally, for the survival selection, usually a generational procedure is used, *i.e.*, the entire parental population is replaced by the newly generated offspring population; another popular method for survival selection in GAs is the steady-state one: each time an offspring is generated, it instantaneously replaces one of the individuals of the parental population according to a given criterium, such as fitness or age.

From this description, it can be seen that GAs are very general methods, with too many degrees of freedom with respect to their parameter choices. We refer the reader to [Whitley, 1994; Mitchell, 1998] for a more comprehensive introduction and an extensive theoretical analysis of GAs as known in the early days.

2.4.5 Differential Evolution

Differential Evolution (DE) [Storn and Price, 1997; Price *et al.*, 2005] is a more recent method proposed to global numerical optimization. As in ES, the solutions are represented by vectors of real-values. The pseudo-code of the standard DE algorithm is shown in Algorithm 2.2, where d is the number of decision variables (also referred to as the dimension of the problem); NP is the population size; F is the mutation scaling factor; CR is the crossover rate; $x_{i,j}$ is the j-th variable of the solution x_i; and u_i is the offspring generated after x_i.

Although achieving very good performance on a wide gamma of problems (see, *e.g.*,

Algorithm 2.2: The Differential Evolution algorithm with DE/rand/1/bin strategy

```
1: Generate the initial population
2: Evaluate the fitness for each individual
3: while the stopping condition is not satisfied do
4:    for i = 1 to NP do
5:       Select uniform randomly r1 ≠ r2 ≠ r3 ≠ i
6:       j_rand = rndint(1, d)
7:       for j = 1 to d do
8:          if rndreal_j[0, 1) < CR or j is equal to j_rand then
9:             u_{i,j} = x_{r1,j} + F · (x_{r2,j} − x_{r3,j})
10:         else
11:            u_{i,j} = x_{i,j}
12:         end if
13:      end for
14:   end for
15:   for i = 1 to NP do
16:      Evaluate the offspring u_i
17:      if f(u_i) is better than or equal to f(x_i) then
18:         Replace x_i with u_i
19:      end if
20:   end for
21: end while
```

all the successful applications listed in [Price *et al.*, 2005]), it is a very simple algorithm To start with, there is no parental selection: each individual in the population is used to generate one offspring; and there is only one (deterministic) method for survival selection: each offspring is compared only with its parent, replacing the parent in case it has a better fitness, sometimes the age of the parent also being considered as a penalty factor.

The generation of an offspring is done by means of mutation and crossover operators. But, differently from the historical convention used in EAs, the mutation takes into account the genetic material of two or more individuals, doing some form of sum of weighted (by the scaling factor F) differences between their components (genes); while the crossover considers just the parent and the intermediary solution generated after the application of the mutation operator, usually referred to as the mutant vector.

Many reproduction schemes have been proposed in the literature, using different mutation and/or crossover operators [Price *et al.*, 2005; Storn and Price, 2008]. In order to distinguish among these schemes, the notation "DE/a/b/c" is commonly used, where "a" specifies the base vector to be mutated; "b" is the number of difference vectors used by the mutation strategy; and "c" denotes the crossover scheme, *binomial* or *exponential*. As an example, in Algorithm 2.2, the reproduction scheme used is the "DE/rand/1/bin" [Price *et al.*, 2005] (see lines 8-12), *i.e.*, the classical "DE/rand/1" mutation strategy, with the binomial crossover.

Other well-known mutation strategies can be listed as follows:

1. "DE/best/1": $\mathbf{v}_i = \mathbf{x}_{best} + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right)$

2. "DE/best/2": $\mathbf{v}_i = \mathbf{x}_{best} + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) + F \cdot \left(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}\right)$
3. "DE/rand/2": $\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) + F \cdot \left(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}\right)$
4. "DE/current-to-best/1"[1]: $\mathbf{v}_i = \mathbf{x}_i + F \cdot \left(\mathbf{x}_{best} - \mathbf{x}_i\right) + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right)$

where $\mathbf{x}_i$ is the current individual or parent, $\mathbf{x}_{best}$ represents the best individual in the current generation, $\mathbf{x}_{r_1}, \ldots, \mathbf{x}_{r_5}$ are different individuals randomly chosen from the current population.

Concerning the crossover operators, the binomial crossover is similar to the uniform crossover used in GAs: for each variable of the problem, the offspring receives the value of the mutant vector with probability CR, from the parent vector otherwise. The exponential crossover is similar to some extent to the GA two-point crossover: components from the parent vector are used up to the first crossover point, randomly selected from $\{1, \ldots, d\}$; then L consecutive components, counted in a circular manner, are copied from the mutant vector, L being a user-defined parameter; and the rest is taken again from the parent vector [Zaharie, 2009]. Although the exponential crossover was used in the seminal DE publication [Storn and Price, 1995], the binomial crossover is much more frequently used nowadays, being said to be "never worse than exponential" [Storn and Price, 2008].

From this description, it becomes clear that one of the main advantages provided by DE is its simplicity when compared to other EAs, as many of its intrinsic procedures are fixed, leaving just three parameters to the user: the population size NP, the mutation scaling factor F and the crossover rate CR, besides the definition of which reproduction scheme to use.

2.5 Application Areas

EAs have been successfully applied to many different application fields, as extensively presented in a recent book completely dedicated to this topic [T. Yu et al., 2008]. Besides the different "flavors" of optimization, which are by far their most important areas of application, EAs have also been used as a "source of creativity" in many other areas. Some of these very diverse application examples will be briefly described in the following.

Combinatorial problems have attracted the attention of EC researchers since the early days of the field, as many important real-world problems can be modeled in this way, what makes it a very profitable area. For example, EAs have been used for diverse *scheduling* problems, such as *crew* and *train* scheduling [Semet and Schoenauer, 2006], task *planning* [Bibai *et al.*, 2010], etc. Most of its success, however, comes from its hybridization with local search and Operational Research (OR) heuristics, as exemplified in the problems of TSP [Merz and Freisleben, 1997], university *time-tabling* [Abdullah *et al.*, 2007] and graph-coloring [Porumbel *et al.*, 2010].

In **continuous** optimization problems, EAs have also greatly shown their value. Specially after the advent of the state-of-the-art, almost parameter-less, CMA-ES technique

[1] "DE/current-to-best" is also referred to as "DE/target-to-best/" or "DE/local-to-best/" [Price *et al.*, 2005; Das *et al.*, 2009].

[Hansen and Ostermeier, 2001], researchers from different domains have been applying it on their very own problems. A very comprehensive and up-to-date list of applications of CMA-ES to continuous optimization problems can be found in [Hansen, 2009b], incredibly counting up to 120 references as of nowadays: very diverse application fields have already been tackled, such as the optimization of gas turbine combustors [Hansen *et al.*, 2009c], or the search for the best craniofacial superimposition for forensic identification [Ibanez *et al.*, 2009]

Another domain of increasing attention, specially since the beginning of this decade, is that of **multi-objective** optimization. By the use of special fitness evaluation and selection methods [Deb, 2001], EAs are known to efficiently find the best set of solutions that satisfies all the objectives under consideration, the so-called *Pareto front*. An interesting application example in this context is that of [Singh, 2006], in which EAs are used to optimize several criteria for the automatic estimation of seismic velocity, a measure used for the possible discovery of petrol in the underground.

Needless to say, the abilities of (i) handling **mixed** search spaces, and (ii) having solutions with **variable-length** (specially true for GP), enable the use of EAs on problems out of reach of standard methods. Besides, as in this way there is no constraint in terms of representation of the solutions, a much more comprehensive (and unbiased) exploration of huge search spaces can be done, possibly leading to the discovery of solutions that could never be imagined by biological intelligence. In this context, several examples of results automatically achieved by EAs that are competitive (and often better) than human performance are presented (and awarded) every year in the so called *Humies* competition [Koza, 2010], sometimes even resulting in patentable products, as presented in [Koza *et al.*, 2000].

For the same reason, EAs have shown interesting results in terms of **creativity** in the *art* and *design* domains [P.J. Bentley et al., 2002; J. Romero et al., 2007], with examples ranging from *architecture* up to *music* automatically generated by evolution. In such kind of applications in which the evaluation of the solutions is "subjective" somehow, a human-in-the-loop is often used to make the role of the fitness function, the so-called Interactive Evolutionary Computation (IEC). In [Quiroz *et al.*, 2007], for example, the IEC paradigm is used for the optimization of user interfaces. One of the main problems of this approach is that of "fatigue": differently from computers, humans do get tired; different proposals have been done in order to reduce such effect, as in [Kamalian *et al.*, 2005].

Lastly, an application domain that is inevitably becoming more and more relevant nowadays is that of sustainable development, where EAs have also shown both their exploration and optimization efficiencies. As combinatorial examples, we can mention the optimization of strategies for pollution prevention [Tan, 2007] and the efficient planning of solid waste management [Yeomans *et al.*, 2003]. It has also been used in the design of "green" buildings, optimizing all the multiple objectives in mixed search spaces that such a project might contain, *e.g.*, the reduction of energy and water consumption, as well as waste and landfill generation [Pitman and King, 2009]. Another example in the same context is the optimization of the topology and parameters of the electronic components that constitute a *Heating, Ventilating and Air-Conditioning* system in order to achieve better energy efficiency [Angelov *et al.*, 2003].

2.6 Discussion

As reviewed in the previous Section, EAs are very general and robust methods, outperforming other approaches and achieving interesting results on very different application domains. It is always unfair, however, to compare their performance with sophisticated problem-tailored methods on the problems to which the latter were specified to: the strength of such special methods is always related to the exploration of problem-specific knowledge, while EAs are general search methods that treat the problem as a black-box function, using the fitness function as the sole source of information. Consequently, the problem-tailored methods, as their name says, perform very well just on their very own problems, while EAs are able to achieve reasonable performance in a much wider range of problems, what might be preferred depending on the situation.

Anyway, given the mentioned characteristics of EAs, the general recommendations for their use can be summarized into the following cases, based in [Eiben, 2002]:

- The search space is very big: in such a case, the brute-force approach becomes prohibitive, consequently turning a (directed) randomized search into a good alternative.

- The search space is mixed, *i.e.*, the variables of the problem have different types (integer and real for example): as discussed in Section 2.3.2, EAs do not have any restriction with respect to the representation of the candidate solutions, whenever corresponding variation operators and initialization procedure are provided.

- The variables of the problem interact with each other in a complex non-linear way, resulting in an objective function with this same characteristic: it is unusual to have specific sophisticated methods for such cases, as extracting some knowledge from the gradient of the search is not a trivial task, while possibly leading to a highly multi-modal fitness landscape (what links to the following item).

- The search space is multi-modal, *i.e.*, with many local optima: the population-based approach employed by EAs enable the exploration in parallel of several promising regions, consequently augmenting the probability of discovering the global optimum; while standard local search methods would tend to get prematurely trapped into local optima. However, whenever a higher (possibly local) optimum is found, the selection pressure will bias the entire population towards it; to avoid this undesired convergence, the maintenance of some diversity into the population needs to be enforced somehow (see, *e.g.*, the niching methods [Horn, 1997]).

- The optimization problem is dynamic, changing over time: the evolution process follows the direction being currently given by the fitness function, no matter if it is static or dynamic, automatically adapting to eventual changes in a transparent way.

- The evaluations are noisy: the evolution is not guided by the evaluation of a single point, but rather by the "trend" gathered from the evaluation of the many points considered in the current population, significantly reducing the noise effect. Besides,

in such cases, several re-evaluations might also be performed until a higher confidence is achieved – indeed, this is a quite common approach, not exclusive to EAs.

It is worthy noting, however, that the use of EAs and other stochastic meta-heuristics do not guarantee the finding of the truly optimal solution for the problem at hand (unless an infinite computational budget is assumed). Accordingly, the fact that more time possibly means the discovery of better solutions explains why one of the main known drawbacks for the use of EAs is their high computational cost. But, no matter the available budget, a solution is always available at the end, be it suboptimal or not; such important property is commonly referred to as *anytime behavior* [Eiben and Smith, 2003].

Another drawback that prevents their broader use is that of parameter setting: the performance of an EA is directly related to how efficiently it *explores* the search space, while also being able to *exploit* the most promising regions. The so-called Exploration versus Exploitation (EvE) balance is controlled by several parameters, that usually need to be defined by the user according to the problem or class of problems at hand. A more comprehensive discussion on parameter setting will be presented in Chapter 3, while Chapter 4 will focus on the parameter setting sub-problem that is addressed by the contributions proposed in this thesis, the Adaptive Operator Selection.

Chapter 3

Parameter Setting in EAs

Contents

In this Chapter, we present a survey about parameter setting in EAs. The possible influence of the setting of some parameters in the EA performance is discussed, together with some propositions for their automatic setting found in the literature. A well-known classification of methods for parameter setting in EAs is also described.

3.1 Introduction

In order to efficiently apply an EA to a given problem, there are several parameters that need to be defined by the user, as surveyed in Chapter 2. In the early days of research in the area, such parameters were seen as an advantage for the EAs, enabling their "personalization" according to the characteristics of the problem at hand.

The optimal values for these parameters were usually defined by intuition, based on rules-of-thumb well known to the community. At the same time, it was believed that researchers would be able to find problem-independent (or universal) "winner" settings, *i.e.*, parameters values that would provide efficient performance to the EAs, independently of the application field. In the context of GAs, two very popular (although very different) "universal settings", published in [Grefenstette, 1986; DeJong and Spears, 1990], are compared in Table 3.1.

	[DeJong and Spears, 1990]	[Grefenstette, 1986]
Population size	50	30
Number of generations	1000	*not specified*
Crossover type	(typically) 2-point	(typically) 2-point
Crossover rate	0.6	0.9
Mutation types	bit-flip	bit-flip
Mutation rate	0.001	0.01

Table 3.1: Two examples of static set of parameters for GAs

However, many further works were published in the following, presenting very different settings for particular problems, consequently putting into question the scientific relevance of the mentioned works. The No Free Lunch (NFL) theorem [Wolpert and Macready, 1997] put an end to this quest for an universal setting, by stating that, roughly, there is no "best algorithm" that solves all problems better than any other. This might be seen as the unique very well-accepted contribution brought by the establishment of the NFL theorem; indeed, it is hardly considered in practice, with several extensions and contradictions having being published since then (see, *e.g.*, [Auger and Teytaud, 2010; Christensen and Oppacher, 2001; Whitley and Watson, 2005]).

Accordingly, the contemporary view of EAs acknowledges that specific (sometimes classes of) problems require specific setups for satisfactory

performance [Eiben *et al.*, 2007]: when it comes to solving a given problem, parameter setting is viewed as the Achilles' heel of EAs, on par with their high computational cost.

This is the main reason to the fact that EAs are very rarely used outside the "evolutionary research labs". Although being tempted by the several empirical demonstrations of the efficiency of EAs on many difficult problems out of reach of other optimization methods, scientists from other domains very often fail in getting interesting results, mainly because of the lack of general methods for tuning at least some of the involved parameters, and also because they are not, and do not want to become, "Evolutionary Engineers".

From these observations, parameter setting in EAs appears actually as a major issue that has deserved much attention during recent years [Eiben *et al.*, 1999], and research is still very active nowadays, as witnessed by a complete edited book that has been recently published [Lobo *et al.*, 2007], and by the numerous recent references cited in this document. These contributions, proposed to automate or guide somehow the definition of the parameters, intend to make the EAs to *Cross the Chasm* [Moore, 1991], enabling the whole scientific community to benefit from their use without their main burden, that of parameter setting.

The following of this Chapter summarizes the research in the field of parameter setting within EAs. Firstly, an overview of the influence of some of the most important parameters on the performance of the EAs, and some examples of what has already been done to automate their setting, are presented in Section 3.2. Then, in Section 3.3, a well-accepted classification of the different ways of doing parameter setting within EAs is described. The Chapter is concluded in Section 3.4, with some further discussions about the topic of parameter setting in EAs. All this material helps into contextualizing the main contributions of this work, the bandit-based methods for Adaptive Operator Selection, described in the following Chapter.

3.2 Parameters Influence and Possible Settings

The EAs presented in Chapter 2 were conceived for different purposes, providing different behaviors. The main difference between them lies in the way the solutions are represented, *e.g.*, standard GAs use binary representation, ESs represent the solutions by vectors of real values, while GP implements tree-based representations. Nowadays, however, original ideas were combined, and it is becoming more and more difficult to differ between EAs. Because of this, [DeJong, 2006] proposes an unified view of such algorithms.

Unified or not, the EAs do have some common structures and parameters, which are independent from the representation being used, as presented in Section 2.3.3. Since the parameter setting techniques presented in the following of this document are not meant to work with just one kind of EA, we briefly discuss here about the influence of these common representation-independent parameters on the search process.

All these parameters affect somehow the Exploration versus Exploitation (EvE) balance: intuitively, as discussed throughout the text, the EA should *explore* the search space in the early stages of evolution, gradually migrating to a more focused *exploitation* of the promising regions. The objective of this Section is not to describe these parameters, as it

was already done in Section 2.3.3, but rather to extend a bit on their influence on the EvE balance, and consequently on the performance of EAs, based on [DeJong, 2007]. Besides, some possibilities of parameter setting are also presented, including a brief bibliographic review for each of them.

3.2.1 Parent and Offspring Population Sizes

A small parent population size m does not enable a good exploration of the (usually very big) search space, possibly converging prematurely to a local optimum; while a bigger m provides a higher probability of finding a global optimum, as multiple peaks might be simultaneously explored. Having a bigger population, however, might slow down the convergence by the fact that more evaluations are needed at each generation. A compromise between both needs to be found.

Following the EvE balance intuition, ideally, the population should be big in the beginning, enabling a better exploration of the search space, with its size decreasing (thus focusing on the most promising regions) as the search goes on. However, according to [DeJong, 2007], the dynamic adaptation of the population size was found to be a difficult task, due to several interacting factors, *e.g.*, selection pressure [Eiben *et al.*, 2006], noisy fitness landscapes [Goldberg *et al.*, 1992], the fact that generations overlap ("plus" replacement) or not ("comma"/generational replacement) [Schwefel, 1995], etc [Arabas *et al.*, 1994; Smith, 1993; Eiben *et al.*, 2004; Bäck *et al.*, 2000].

Still from the EvE balance point-of-view, the parent population represents which regions of the search space are being currently explored, while the ratio between its size m and the offspring population size n defines the amount of exploration done by it at each generation. The ideas and methods proposed for the adaptation of the parent population size are also valid for the size of the offspring population; we refer the reader to [Jansen *et al.*, 2005] for a comprehensive analysis and some specific proposals for the dynamic adaptation of this parameter.

3.2.2 Selection Procedures

As previously presented in Sections 2.3.3 and 2.3.3, there exist several families of selection procedures for both, the parent selection *pselect* and the replacement or survivors selection *sselect* (see Section 2.3.3), *e.g.*, fitness-proportional (or roulette-wheel) selection, rank-based selection, tournament selection. Defining which of them to use within an EA for the problem at hand is already a complex choice; in addition, there are still the sub-parameters to be tuned.

As for the other parameters, the migration from exploration to exploitation is related to the level of selection pressure that is exerted, *i.e.*, less selection, more exploration, and vice-versa. In the case of the standard tournament selection, for example, as described in Section 2.3.3, the smaller the tournament size T (*i.e.*, the less individuals are chosen to participate in the tournament), the more random the selection is; oppositely, higher is the T, higher is the chance of the better individuals being considered, consequently more

elitist is the selection process. Along the same lines, the probability t of selecting the best individual between the two chosen individuals in the stochastic tournament selection can range from a totally random selection ($t = 0.5$) to a completely elitist strategy ($t = 1$).

However, the control of such selection pressure is not ruled simply by the setting of these parameters; indeed, it is defined by the combined effects of both parent and replacement selection procedures, not mentioning other interacting effects, such as the population size [Eiben *et al.*, 2006]. This complexity might be the reason why so few references can be found on the dynamic adaptation of the selection pressure. In [Herrera and Lozano, 1998], a fuzzy model is used to automatically control the tournament size, based on the genotypic and phenotypic diversity measures; more recent works propose [Eiben *et al.*, 2006] and better empirically validate [Vajda *et al.*, 2008] a hybrid self-adaptive tournament size which achieves much better results than the fuzzy model, being referred to as hybrid by the fact that the parameter control is done by a combination of self-adaptation and feedback-based or adaptive control (see Section 3.3.2 for a brief overview of the different ways of doing parameter control). After the proof-of-principle presented on these latter references, the use of parameter control for selection methods has shown to be a viable path to be further explored towards more efficient and easier-to-tune EAs [Vajda *et al.*, 2008].

3.2.3 Offspring Production

The procedure for offspring production, *reprod*, can be classified as representation-dependent, as the variation operators need to be defined according to the representation being used in order to be able to generate feasible solutions. The application of these operators directly impact the EvE balance, consequently affecting the effects provided by all the previously mentioned parameters [DeJong, 2007], as follows: while the selection pressure tends to reduce the population diversity, variation operators are responsible for counter-balancing this effect by, as their name says, introducing variation into the population; the quantity of novelty to be possibly introduced, however, depends on the population size and on the level of diversity in the current population.

Such correlation between the parameters makes it very complex to decide which operators should be included in the EA algorithmic framework for a given problem, and how to set their sub-parameters. Adding this possible correlation to the stochastic nature of the underlying algorithm, it becomes very difficult to know a priori how a given operator (with a given configuration) will behave during the search process; besides, different operators (or different configurations of the same operator) might perform differently at different stages of the search, according to the characteristics of the region of the fitness landscape currently being explored by the population.

An alternative to take this decision out of the user's burden is to automatically adapt the internal parameters of a given operator. In the context of ESs, for example, the variance of the gaussian mutation operator has been automatically adapted since the early days, starting with the "1/5th success rule" [Schwefel, 1975], until the advent of the very popular and current state-of-the-art CMA-ES [Hansen and Ostermeier, 2001]. This latter is, indeed, the most (and almost unique) successful case of a parameter control technique within EAs to date, after more than 30 years of research in the area [DeJong, 2007].

Another plausible approach is to maintain a collection of operators, and to dynamically select the ones that are affecting the search process in a more beneficial way [DeJong, 2007]. The selection of which operator among the several available operators should be used, what we here refer to as Adaptive Operator Selection (AOS), is representation-independent. Accordingly, the AOS methods proposed in this thesis can be applied to any of the existent or newly proposed EAs[1] – as a representative set, in Chapter 6 we show their use within GAs and DE. Following the intuition, ideally, the dynamic selection of operators should promote the use of the more explorative operators in the beginning, preferring the less perturbative ones (exploitation) in the later stages of the search. An extensive bibliographic review on this, which is the central topic of this thesis, is presented in Chapter 4.

3.2.4 Stopping Criterion

The stopping conditions do not affect the EvE balance; indeed, they define how the computational budget is spent, as discussed earlier in Section 2.3.3.

This is a clear example of a representation-independent component that could be defined in a more autonomous way, although depending on a representation-based criterion, the population diversity. Anyway, works proposing the dynamic adaptation of this parameter were not found in the literature; fixed strategies are commonly used, being defined after an expensive off-line tuning phase (see Section 3.3.2) or more frequently via intuition and/or budget constraints.

3.2.5 Representation

One of the main choices is very probably how to represent (and consequently manipulate) the candidate solutions of a given problem. Although greatly affecting the performance of EAs, the representation is very often defined a priori, guided by a large body of literature [DeJong, 2007]. Such definition is often static, with very few works considering its dynamic adaptation during the search process.

The effects of the adaptation of the representation can be said to be two-fold. On the one hand, it can be used to improve the effectiveness of operators, by adapting the representation according to the characteristics of the operator, like in the Messy-GAs [Goldberg *et al.*, 1991], in which the position of the genes on the chromosome are constantly modified while solving the problem so that the 1-point crossover operator maintains its good performance throughout the search process. On the other hand, it can also be used to bring (or contribute into maintaining) invariance properties to the EAs, as in the recent Adaptive Encoding method [Hansen, 2008], which provides, to any continuous search algorithm, invariance with respect to rotation over a given problem function, based on the CMA-ES.

[1] Indeed, the proposed AOS techniques can also be extended to other local search heuristics, but this discussion is out of the scope of the current Section.

3.3 Classification of Parameter Setting Techniques

Very different parameter setting methods have already shown their usefulness in the literature by automatically setting representation-independent and also algorithm-specific parameters of EAs. A classification of these techniques, proposed in [Eiben *et al.*, 1999], and later revised in [Eiben *et al.*, 2007], is very well-accepted by the community, as acknowledged by the number of citations it received. Since it is used to classify the methods proposed in this work, it is reminded in the following, for the sake of self-containedness.

It categorizes the parameter setting methods according to four different aspects, listed as follows: (i) Which parameter is changed? (ii) How the changes are made? And more precisely for the parameter control techniques: (iii) Which evidences guide the changes? (iv) And which is the scope of the change? Each of these aspects will be briefly discussed in the following.

3.3.1 Which parameter is changed?

The first criterion adopted for the classification concerns which component or parameter of the EA is being changed. Although there is no standard list of components, we consider here the components described in Section 3.2.

As already mentioned, each of the listed parameters might also have some sub-parameters, *e.g.*, number of bits to be flipped by the bit-flip mutation operator, tournament size for the tournament selection, etc. However, the objective of this classification is rather to be able to easily locate, within the standard EA loop, which steps are affected (hopefully improved) by the proposed changes.

The Adaptive Operator Selection techniques proposed in this work provide to the user an autonomous control of the use of the available variation operators, which can be seen as an adaptation of their application rates (despite the fact that the proposed bandit-based techniques, presented in Chapter 5, do not rely on probabilities for the operator selection).

3.3.2 How the changes are made?

The changes in the parameter values can be made, mainly, in two different ways, as illustrated in Fig. 3.1: before the main run of the algorithm on the given problem, referred to as off-line or external **parameter tuning**; or during the run, while solving the problem, referred to as on-line or internal **parameter control**. A brief description about each of them is presented in the following.

Off-line or External Parameter Tuning

Methods that perform off-line or external tuning determine *a priori* the appropriate parameter values, based on the results of several runs of this algorithm. The algorithm to be tuned is usually considered as a black box, with the tuning method guiding the exploration of the search space of the parameter values. Off-line tuning methods can be further subdivided into two main classes: pure statistical methods, and optimization methods, which

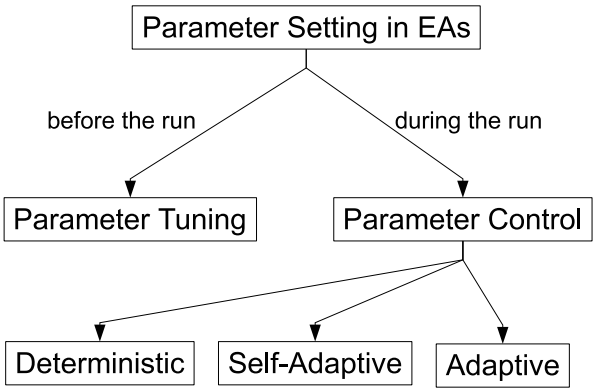


Figure 3.1: Classification of parameter setting methods, from [Eiben *et al.*, 1999].

treat the parameter tuning as an optimization problem itself; some prominent examples for each of them are briefly described in the following.

Starting with the statistical methods, the most basic (and computationally expensive) way of doing so lies in the execution of a complete Design of Experiments (DoE) process, which can also be referred to as a factorial design, or even a brute-force approach: the parameter values are discretized into m candidate configurations, each of them is independently assessed n times, and the best configuration is extracted according to some *ANOVA*-like statistical test over this $m \times n$ performance data. In practice, it becomes very computationally expensive to tune even just a few parameters, *e.g.*, by considering just 4 parameters, each parameter with 5 possible values, will already lead to $5^4 = 625$ candidate configurations to try out (not considering possible cross-influences between parameters).

The Racing techniques [Birattari *et al.*, 2002; Yuan and Gallagher, 2004] do basically the same, but in a much less time-consuming way, as follows. As in the DoE, the parameter values are also discretized into m candidate configurations. But, as soon as a candidate configuration is statistically found to be significantly worst than the current best configuration (after some runs, depending on the variance of the achieved results), there is no need to keep further assessing it; this configuration is thus eliminated from the tuning process. In this way, the computational resources are more efficiently used, focusing just on the most promising candidate configurations, consequently leading to lower variance performance estimates for them. The use of this approach results into important time savings, as illustrated in Fig. 3.2. The x-axis Θ represents the number of remaining candidate configurations, and the y-axis "i" shows the number of evaluations or "racing laps" done for each of them; the amount of computation needed for both, the F-Race [Birattari *et al.*, 2002] and the brute-force approaches, are represented by the areas of their respective surfaces.

A prominent example of Racing techniques is the F-Race [Birattari *et al.*, 2002], which uses the "Friedman's two-way analysis of variance by ranks" as statistical test to eliminate candidate configurations. This is the method used to tune all the hyper-parameters of the proposed and baseline AOS techniques for the empirical comparisons presented in Chapter 6, as described in Section 6.2.2.

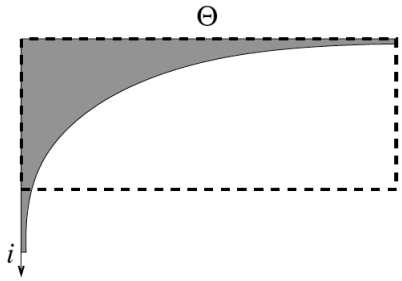


Figure 3.2: A visual representation comparing the amount of computation needed by the brute-force approach (dashed rectangle) and the F-Race method (shadowed area), reproduced from [Birattari *et al.*, 2002].

But, although saving a significative amount of computational budget, the use of the F-Race technique can become computationally prohibitive whenever there is a large number of parameters and a wide range of possible values for each parameter, as some initial runs need to be done for each candidate configuration before the first elimination round. A very simple alternative proposed to this problem is the use of a sampling of the whole set of configurations [Balaprakash *et al.*, 2007]. In case a priori knowledge about the configurations search space is available, it can be used to define the probabilities of sampling each configuration; however, as this is usually not the case (and remembering that a priori information might also include a wrong bias in the search, as discussed in Section 2.3.2), the authors propose the use of a completely random sampling of the configurations. The resulting method is referred to as *Random Sampling Design F-Race* (RSD/F-Race) [Balaprakash *et al.*, 2007].

An alternative to the parameter tuning problem, as previously mentioned, is to consider it as an optimization problem on its own, *i.e.*, by varying the parameter values, the objective is usually to optimize some measure, such as the performance of the algorithm over a given problem or class of problems, or its robustness with respect to several problems, etc. Based on this, it becomes straightforward to think about the use of optimization methods for this task, thus at a higher level of abstraction, commonly referred to as the "meta" or "hyper" level. EAs themselves have already been used to do so, defining the so-called *Meta-EAs* [Clune *et al.*, 2005; Yuan and Gallagher, 2007]. The problem in this case lies in how to define the parameters of the EA in the meta-level.

The ParamILS [Hutter *et al.*, 2009] method uses an iterated local search algorithm to explore the neighborhood of the best parameter values found so far, using some random perturbations and restarting the search from time to time (according to a user-defined probability) to enforce a better coverage of the search space. Its very general idea, combined with an adaptive limiting of the time spent for evaluating individual configurations, enables it to be used on very different situations: indeed, it was already shown to efficiently tune algorithms with up to 10^{37} configurations [Hutter *et al.*, 2009].

Along the same lines, another popular optimization method used to off-line tuning, the Sequential Parameter Optimization (SPO) [Bartz-Beielstein *et al.*, 2005], combines

modern statistical approaches for deterministic algorithms as the Design and Analysis of Computer Experiments (DACE) with classical regression techniques in order to tune stochastic algorithms such as EAs. The set of candidate configurations being assessed is constantly refined during the tuning procedure, what is done by means of Gaussian processes, with some configurations being eliminated and new ones being inserted in the pool according to the current model of the parameter space. At a higher level of abstraction, rather than simply a tuning method, the SPO can also be seen as a methodology for the empirical analysis of stochastic optimization algorithms, providing to the experimenter a very well-defined twelve-step procedure.

Another model-based optimization method applied to parameter tuning is the Iterated F-Race (I/F-Race) [Balaprakash *et al.*, 2007; Birattari *et al.*, 2009], which is yet another improved variant of the F-Race, more complex although more efficient than the RSD/F-Race approach. Starting from the initial set of possible parameter values for each parameter (as done in the original F-Race), at each iteration, some efficient configurations are used to update a probabilistic model about the configurations search space. This model is then used to generate new candidates, consequently biasing the search towards the most promising parameter configurations.

A very different approach is proposed by the Relevance Estimation and Value Calibration (REVAC) method [Nannen and Eiben, 2007]. It uses Shannon and differential entropy in order to find parameters with higher impact on the performance of the algorithm, while also estimating the utility of their possible values. Thus, besides tuning the parameters of the algorithm, it provides to the user a high-level information about their relevance, which can in turn be used in order to better allocate the resources for their calibration, *e.g.*, by providing more resources for the tuning of the most important or sensitive parameters.

All these methods have already proved their efficiency and usefulness in different ways in the literature. An advantage provided by them is that, as they use just generic performance measures, they are not limited to EAs, being possibly applied to many other stochastic algorithms, while being also very easily combined; in [Smit and Eiben, 2009], *e.g.*, an extensive empirical comparison is presented between different pure and hybrid off-line tuning methods, including meta-EAs, REVAC, SPO and Racing. However, given the stochastic nature of EAs, each performance assessment corresponds in fact to the average of a few evolutionary runs, what makes the off-line tuning a very expensive procedure. Furthermore, static settings are usually provided by these methods (the parameter value is fixed along the run), whereas the optimal setting likely depends on the local landscape being explored by the population, as previously discussed.

On-line or Internal Parameter Control

Internal **parameter control** methods work directly on the values of the parameters while solving the problem, *i.e.*, on-line. Such kind of mechanisms for modifying parameters during an algorithm execution were invented early in EC history, and most of them are still being investigated nowadays. Indeed, there is at least two strong arguments to support the idea of changing the parameters during an EA run:

- As evolution proceeds, more information about the algorithm behavior within the current fitness landscape is known, so it should be possible to take advantage of it. This applies to global (for example, knowing how rugged is the landscape) and to local properties (for example, knowing whether a solution has been improved lately or not).
- As the algorithm proceeds from a global (early) exploration of the landscape to a more focused, exploitation-like behavior, the parameters should be adjusted to take care of this new reality. This is quite obvious, and it has been empirically and theoretically demonstrated that different values of parameters might be optimal at different stages of the search process (see [Eiben *et al.*, 2007] and references therein).

The different approaches that have been proposed to internally adapt the parameters can be gathered into three categories, depending on the type of information used for the adjustment of the parameters values, as presented in Fig. 3.1 [Eiben *et al.*, 2007]. Each category is briefly reminded in the following, including some examples in the context of variation operators adaptation.

Deterministic parameter control implements a set of deterministic rules without any feedback from the search. This is, in general, hard to achieve, because of a simple reason: it relies heavily on knowing beforehand how long the EA will take to achieve a given target solution with the running algorithm, what can not be easily predicted. But even if it were, the way to balance exploration and exploitation can hardly be guessed. This situation is worsened by two facts: first, given the stochastic nature of EAs, there is usually a big variance between different runs *on the very same problem*; and second, these methods often require additional parameters that are used to tune the deterministic parameter itself (starting with the total number of generations the algorithm will run), and even though these parameters can be considered second-order, their influence is nevertheless critical. Given these difficulties, these methods were mainly used in the early days of research in the area, as in [Hesser and Männer, 1990], in which a theoretically optimal schedule was proposed to deterministically adapt the mutation application rate, based on the elapsed number of generations.

Since our knowledge about the way the search should behave is always limited, it is sometimes possible, and advantageous, to let evolution itself tune some of the parameters: this kind of parameter control approach, referred to as **Self-Adaptive**, adjusts parameters "for free", *i.e.*, without any direct specification of the user. In other words, individuals in the population might contain "regulatory genes" that control some of the parameters, *e.g.*, the mutation and recombination rates; and these regulatory genes would be subject to the same evolutionary processes as the rest of the genome [DeJong, 2007]. During quite some time in the 90s, self-adaptation was considered as the royal road to success in Evolutionary Computation. First of all, the idea that the parameters are adapted *for free* is very appealing, and the parallel with self-regulated genes is another suggestive argument. On the practical side, as self-adaptive methods require little knowledge about the problem and, what is probably more important, about the way the search should proceed, it

sometimes remains the only way to go when nothing is actually known about the problem at hand. As an early example, in [Bäck, 1992], each individual's representation was extended by 20 additional bits, which were used to encode its own self-adapted mutation rate, a real value between 0 and 0.5. In [Spears, 1995], a single bit was used to represent which of two crossover operators (uniform or 2-points) should be applied, resulting in a much better performance than the one achieved by the static use of a single operator. A very recent and comprehensive review of the current state of research on self-adaptation methods, with special hints for its use within combinatorial problems, can be found in [Smith, 2008]. Although having shown to be an efficient approach in many different situations, the main drawback is that the algorithm needs to explore, in parallel, the search space of the variables of the problem and also the search space of the parameter values, what potentially increases the complexity of the search.

Then, it becomes clear that whenever some decisions can be made to help the search following an efficient path, this should be done. **Adaptive** or **Feedback-based** methods follow this rationale, being the most successful approaches nowadays in on-line parameter tuning. These methods are based on the monitoring of particular properties of the search/optimization process, and use changes in these properties as an input signal to change the parameter values. The most prominent example of adaptive methods, and one of the main very well-established successful stories of automatic parameter control within EAs, is that of CMA-ES [Hansen and Ostermeier, 2001], in which informations about the gradient and the trajectory of the search are used to automatically adapt the step-size and the shape of the ES mutation operator. Lots of achievements have been being reported also by the Operational Research and Local Search communities, in which the adaptive methods are referred to as "Reactive Search"; a recent survey on this can be found in [Battiti *et al.*, 2008]. The main contributions of this work, the Adaptive Operator Selection (AOS) methods proposed in Chapter 5, are included in this latter category. Besides, a comprehensive bibliographic review on the topic, with several other examples, is presented in Chapter 4.

3.3.3 Which evidences guide the changes?

When using adaptive parameter control techniques, the parameter values are adapted based on the monitoring of some measures of the progress of the search. A further criterion commonly used to classify these techniques is the kind of evidences which are used to guide these changes done while solving the problem [Smith, 1998]. Using as example the AOS techniques proposed in the following of this work, this feedback from the search progress can be provided in two different ways, as follows.

The most common *Credit Assignment* scheme for AOS considers the real values of the fitness improvements achieved by the application of each operator. Starting from the common Instantaneous and Average values, up to the use of Extreme values supported by us (see Section 5.2.2), all of them reward the operator (and consequently guide the changes in the operators preferences) based on raw values, which are **absolute** evidences.

Differently, the *Credit Assignment* that use ranks over the raw values of the fitness

improvements, as presented in Section 5.2.4, control the choice of operators based on the ranking of the same fitness improvements. Thus, it is not the magnitude of the improvement brought by the application of the operator that matters, but rather how good it is with respect to the others, what is referred to as **relative** evidence.

3.3.4 Which is the scope of the change?

Yet another aspect used to classify parameter control techniques lies in the scope of the adaptation being done. According to [Angeline, 1995], the adaptation might happen at the level of the **population**, the **individual**, or the **component**. This factor is not only related to the parameter control method itself, but also to the parameter that is being adapted, as acknowledged in [Eiben *et al.*, 2007].

At the *population* level there are the methods that adapt *global* parameters, *i.e.*, parameters affecting the whole population. A very early example of adaptive method at the population level is the so-called "1/5th success rule" [Schwefel, 1975]: if the applications of the mutation operator succeed in generating offsprings that are fitter than their respective parents in more than 1/5 of the trials, the mutation step-size should be increased by an user-defined fixed ratio, decreased otherwise. The recent state-of-the-art CMA-ES [Hansen and Ostermeier, 2001], which adapts the step-size and the "shape" (defined by a Hessian matrix) of the mutation operator based on the results of its latest applications is another example of an adaptive method affecting the whole population. The AOS techniques presented in this thesis also act at the population level, adapting the operators application rates which are globally used for the generation of every new candidate solution. Rather than adapting somehow the operators application step, in [Eiben and van Hemert, 1999] the SAW method is proposed to globally adapt the fitness function for constraint satisfaction problems, what is done by dynamically changing the weight of each gene (representing a constraint) on the fitness function, with the harder constraints affecting more highly the fitness evaluation, consequently resulting in a higher reward for the creation of individuals that succeed in satisfying them.

The methods at the *individual* level control parameters that *locally* affect each individual. As an example, the self-adaptive methods that encode the (GAs) operators application rates [Spears, 1995] or the (ESs) mutation step-size [Beyer, 1995], within the genotype of each individual, affect just the candidate solution to which they are attached to. A recent example of adaptive method that affects each solution individually is the Multi-Objective (MO) CMA-ES [Igel *et al.*, 2007]: briefly, the same adaptation implemented by the original CMA-ES is used to adapt the mutation operator carried by each individual, whenever its application is successful, *i.e.*, whenever it succeeds in generating a fitter offspring.

At the lowest level of abstraction considered here, there are also methods that adapt parameters within an individual, at the so-called component level. Exemplifying, in [Schwefel, 1995] a self-adaptive ES was proposed, in which each element of the real-valued vector representation has a variance parameter attached to it; thus, each gene of the individual is mutated according to its own self-adapted variance parameter.

The **hyper** level could also be considered here, as recommended in [Maturana, 2009], aggregating the recent methods that control the usage of several heuristics in different

ways, referred to as *hyper-heuristics* [Burke *et al.*, 2010].

3.4 Discussion

As discussed throughout this Chapter, the performance of the EAs is directly related to how the Exploration versus Exploitation (EvE) balance is addressed by the algorithm: if too much exploration is done, the search will very probably take too long to achieve the optimum; while if too much exploitation is done, the search is very likely to prematurely converge to a local optimum. To achieve an acceptable performance, a compromise between both terms needs to be found, what is controlled by some of the EA parameters, as described in Section 3.2.

After the No Free Lunch theorem [Wolpert and Macready, 1997], it is acknowledged that there is no algorithm that performs best over all optimization problems. Considering two instances of the same EA with different parameter settings as two different algorithms, it states thus that there is no winner universal parameter setting, *i.e.*, specific problems require specific parameter settings. These findings lead to a kind of *dilemma*: in order to manually setup the parameters of the EA according to the problem at hand, the user would need to analyze and understand the characteristics of the problem; while one of the main reasons for the use of EAs and other meta-heuristics is, indeed, the lack of knowledge about the problem. Avoiding such dilemma, by automating the task of parameter setting, is thus one of the main motivations for research on this topic.

After the analysis of the influence of the parameters in the performance of EAs, however, it can be said that almost all of them affect the EvE balance somehow, with one parameter counter-balancing or intensifying the effects of the other. This interaction makes more complex the task of automatic parameter setting, being one of the main reasons why just very few works try to address more than one parameter at the same time.

Parameter setting in EAs can be done mainly in two different ways, as described in Section 3.3. Off-line parameter tuning methods consider the EA as a black-box, using just the performance of the algorithm (usually averaged over several runs given its stochastic nature) in order to choose the best set of parameters, which are usually "statically" used during the whole search process; while, intuitively, the EvE balance of the algorithm should be continuously modified while solving the problem, gradually switching between exploration and exploitation according to the progress of the search. This on-line dynamic adaptation is what is provided by the so-called parameter control methods, category in which the contributions proposed in this work are included. From these observations, it is important to note that, even in the case of static problems, they become dynamic from the point of view of the parameter setting task, thus being another motivation for the use of dynamic strategies. Needless to say, an even higher payoff might be achieved in the cases in which the problem itself is dynamic, with its fitness landscape changing over time [DeJong, 2007].

The methods proposed for the automatic parameter setting, however, present their own parameters that also need to be defined by the user, referred to as *hyper*-parameters in the following of this text. Although it might seem not so interesting to replace some

parameters by others, the hyper-parameters are at a higher level of abstraction, being thus (ideally) more easily "understood" by the user and less sensitive than the original EA parameters with respect to their tuning. For example, in the case of the Adaptive Operator Selection techniques proposed in this work, described in Chapter 5, two or three hyper-parameters (depending on the method) need to be configured, while in the original EA framework the user would need to define a very complex and problem-specific scheduler in order to have the same kind of adaptive behavior. These hyper-parameters can then be tuned by off-line tuning methods, as done for the experimental comparison presented in Chapter 6; or extra layers of parameter control could be added, what is always worthy whenever the assumptions about the easier comprehension and smaller sensitivity of the higher-level parameters with respect to the lower-level ones are held.

Although not being part of the scope of this thesis, another viable path for the parameter setting in EAs would be to try to build a knowledge base correlating somehow the parameters of the problem instances solved (the so-called problem descriptors) with the respective parameters used by the algorithm to achieve good performance. In this way, after logging data from a "sufficient" amount of instances and parameters, whenever a new instance that needs to be solved is recognized as part of a certain class of problems, the parameter values that were already optimized to a previously seen instance of the same class can be re-used, thus no need of further tuning it. Such kind of approach has been successfully applied in the domain of SAT heuristics and problems (see, *e.g.*,[Hutter and Hamadi, 2005]); however, to the best of our knowledge, there does not exist yet a well-established set of descriptors for the kind of instances commonly tackled by EAs, being thus a possible path for further research.

Chapter 4

Adaptive Operator Selection

Contents

In this Chapter, we focus on the parameter setting problem addressed by our contributions, referred to as Adaptive Operator Selection (AOS). The components needed to do AOS, namely, the Operator Selection and the Credit Assignment, are described, and some examples found in the literature are surveyed.

4.1 Introduction

In order to efficiently apply an EA to a given problem, there are commonly two design choices that need to be done by the user concerning the variation operators: (i) which of the existent operators should be used by the evolutionary scheme for the generation of new solutions, and (ii) at which rate each of the chosen operators should be applied. As discussed in Section 2.3.2, there are different kinds of operators for each representation scheme, namely mutation and crossover operators (not mentioning the problem-specific and/or the local search operators). Each one of them has its own characteristics, affecting the Exploration versus Exploitation (EvE) balance in its own manner, as also briefly discussed in Section 3.2.3. This scenario makes these operator-related choices very sensitive and complex, as follows.

First of all, the performance of a given operator usually depends on the characteristics of the **problem** being solved. Since it is very difficult to foresee a priori how well a given operator will perform on the problem at hand, the natural choice in this sense would be to use an off-line tuning technique, such as the ones surveyed in Section 3.3.2, in order to find out which operator(s) should be used, and how. Although being very computationally expensive, these off-line methods usually succeed in providing to the user the best static strategy, consisting of one or a few operators that are applied at fixed rates during the whole search process.

However, the performance of the operators in fact does not solely depend on the global characteristics of the problem, but rather on the local characteristics of the **region** of the search space that is being currently explored by the population, which can be more adapted or not to the characteristics of the operator. Finally, their performance also depends on the **state** of the search process, *i.e.*, if it is approaching or not the optimum, how diverse the population is, etc. For example, following the very basic intuition of the EvE balance, already discussed in Section 3.2.3, more exploratory operators might achieve better performance in the early stages of the search, while more exploitation-like/fine-tuning operators might bring better improvements to the search when it is getting closer to the optimum. Based on these issues and on the stochastic nature of the underlying algorithms (one run might be very different from another on the very same problem), the static strategies provided by the off-line tuning methods tend to perform sub-optimally: ideally, the choice of the best operator to be applied should be continuously adapted while solving the problem, *i.e.*, in an on-line fashion.

On-line parameter setting methods are commonly referred to as *Parameter Control* [Eiben *et al.*, 2007]; there are different ways of dynamically doing so, namely, in a self-

adaptive or in an adaptive manner, as reviewed in Section 3.3.2. The self-adaptive methods have the advantage of tuning the parameters "for free", by the evolution itself, adapting the best operator according to the region being "locally" explored by each individual solution; however, besides augmenting the overall complexity of the problem to be solved by aggregating the solutions search space with the parameters search space, these methods are intrinsically linked to the EA structure. Oppositely, the adaptive methods might be more complex to implement, while presenting a few hyper-parameters that also need to be tuned; but they consider the problem search space as it is; and since the adaptation of the parameters is usually guided by general assessments of search progress, the adaptive methods methods can be easily extended to other meta-heuristics and/or stochastic local search methods.

4.2 Adaptive Operator Selection

Based on all the above arguments, we have decided to tackle the operator selection problem with adaptive parameter control methods, thus aiming at the on-line selection of the best operator, *i.e.*, while solving the problem. We refer to such methods as Adaptive Operator Selection (AOS). Fig. 4.1 depicts a high-level view of how AOS methods can be integrated within an EA, which can be read in a general way as follows.

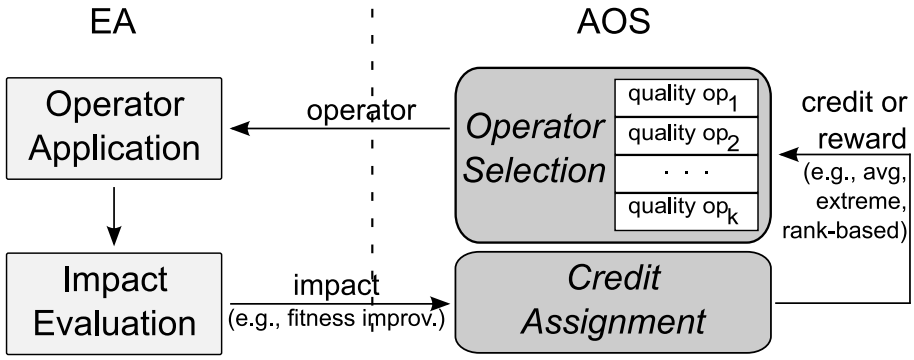


Figure 4.1: The Adaptive Operator Selection general scheme.

1. For the generation of each new trial solution (or after n trials or generations), the EA asks the AOS which of the available operators should be applied.
2. The AOS returns the operator to be used, according to its ***Operator Selection*** mechanism, which selects one operator based on the recent performances of all operators, usually represented by an estimate of their empirical qualities, as discussed in Section 4.4.
3. The selected operator is applied by the EA, a new solution is generated, consequently impacting somehow the search, *e.g.*, generating an offspring better than its parent (fitness improvement), varying the mean diversity of the population, etc, as surveyed in Section 4.3.

4. This impact assessment is transformed into a credit (also referred to as reward), according to the implemented ***Credit Assignment*** scheme.

5. This credit or reward is then used to update the empirical quality (or performance) estimates kept for each operator by the *Operator Selection* scheme, which will be used the next time it needs to select one of the operators.

6. This loop happens continuously while solving the problem, in a on-the-fly reinforcement learning fashion.

As can be seen from this description, designing an AOS method requires the definition of two components: (i) the *Credit Assignment* scheme, that assigns credit to an operator based on the impact brought by its recent application on the current search/optimization process; and (ii) the *Operator Selection* mechanism, which selects the next operator to be applied, based on the knowledge built by the stream of these empirical assessments. In the following, these components will be separately discussed, and some existing methods will be surveyed.

4.3 Credit Assignment

Several *Credit Assignment* mechanisms have been proposed in the literature, following Davis' seminal paper [Davis, 1989], differing mainly on three aspects: (i) how the impact of the operator application should be measured; (ii) how to assign credit based on these impact assessments; and (iii) to which operator(s) the credit should be assigned to. Each one of these aspects will be briefly detailed and exemplified, respectively, in Sections 4.3.1 to 4.3.3.

Finally, although the most common impact measure is the fitness improvement, diversity becomes important as well when tackling multi-modal problems. The *Compass* [Maturana and Saubion, 2008a], which is a method to aggregate both measures, is used in our experimental section, within a GA applied to SAT problems (see Section **??**). For the sake of self-containedness, the *Compass* method will be described in Section 4.3.4.

4.3.1 How to measure the Impact?

In order to measure the impact of the application of an operator on the search process, most approaches consider the improvement achieved by the fitness of the generated offspring with respect to a reference value. This reference might be a local value (*e.g.*, the fitness of its parents [Tuson and Ross, 1998; Wong *et al.*, 2003; Ho *et al.*, 1999]), or a global/population-based value (such as the fitness of the current best individual [Davis, 1989; Lobo and Goldberg, 1997], or the median or some other quantile fitness [Julstrom, 1995; Julstrom, 1997]). In [Barbosa and Sá, 2000], an aggregation of both local fitness improvement and global improvement (with respect to the 90% quantile of the current fitness distribution) is used to assess the productivity of the operators.

Instead of directly using the raw values of the fitness improvements to assess the impact, other approaches measure a relative value. For instance, in [Giger *et al.*, 2007],

the improvement of the offspring with respect to the parent is divided by the gap between its fitness and the fitness of the current best operator (in the case of minimization; the inverse otherwise). Other authors consider a much simpler version of impact measure: the fact that the operator application was successful or not. A successful application means that the generated offspring has a better fitness than its reference value. In [Niehaus and Banzhaf, 2001], the success over the parents is considered; while in [Luchian and Gheorghies, 2003], the success over the best, the success over the parents, plateau walks (same fitness than its parents) and worsenings (fitness lower than its parents), are all aggregated in order to accurately characterize the impact of an operator application. Although not being explicitly mentioned, in [Julstrom, 1995; Julstrom, 1997] only the measure of success is used, resulting in a 1 whenever an improved offspring is generated, 0 otherwise.

Different measures, such as the rank of the offspring within the current population, or the age of the solution in number of generations (in this case the adaptation happening only every n generations) can also be found in [Whitacre *et al.*, 2006]. In most approaches, when there is no improvement, the offspring is simply discarded or, most commonly, the operator application is evaluated as having a null impact. This latter choice is the one employed by all AOS techniques developed in this work, unless stated otherwise.

In the case of multi-modal optimization, another relevant impact measure concerns the population diversity; a minimal level of diversity should be enforced in order to avoid premature convergence. To measure diversity, the Hamming or the Euclidean distances are commonly used. In [Giger *et al.*, 2007], the relative fitness improvements and the mean Euclidean distance are independently used, depending on the needs of the search with respect to exploitation or exploration. Along the same lines, [Maturana and Saubion, 2008a] proposed an impact measure called *Compass*, defined as a weighted sum of fitness improvement and mean diversity (Hamming distance) variation (see Section 4.3.4 for more details). In [Maturana *et al.*, 2010b], two different aggregation methods considering both impact measures were proposed, based directly on the *Pareto Dominance* paradigm.

4.3.2 How to assign Credit?

Based on the impact measures received, at some point a credit needs to be assigned to the operators, in order to update the empirical quality estimates about their performance. These estimates are then used by the *Operator Selection* mechanism under employment (see Sections 4.4 and 5.3 for a few examples), the next time it needs to select one of the operators to be applied, as discussed in Section 4.2.

This credit assigned to the operator is defined after its impact in the progress of the search. It can be the instantaneous value, *i.e.*, the impact measure received after its most recent application; but this tends to be very unstable and noisy, given the stochastic nature of the underlying algorithm: an operator might just have been unlucky on its latest trial, and this will erroneously reflect on the update of its application rate. This is often remedied by an aggregation of credits in the *Operator Selection* side, as done in [Lobo and Goldberg, 1997; Barbosa and Sá, 2000].

A more robust, and by far the most common approach is to use as credit the av-

erage of the latest W applications, W being the size of the sliding time window. The impact measures of the operator are hence aggregated over a given time period, as done in [Davis, 1989; Julstrom, 1995; Julstrom, 1997; Ho *et al.*, 1999; Wong *et al.*, 2003; Giger *et al.*, 2007; Maturana and Saubion, 2008a; Maturana *et al.*, 2010b]. In case the impact measure being used is the success, *i.e.*, 0 or 1 depending if it succeeded in generating a fitter offspring or not, the average is usually used, being simply referred to as the success rate [Niehaus and Banzhaf, 2001; Luchian and Gheorghies, 2003] of the operator. Though the instantaneous version can be viewed as an average over a window of size 1, both will be distinguished in the remainder of this text, termed respectively *Instantaneous* and *Average Credit Assignment* schemes.

A very different approach is the one proposed in [Whitacre *et al.*, 2006], which assigns credit to the operators based on their ability to generate outlier solutions, following some statistics over the received impact measures. The underlying idea is that the generation of rare but highly beneficial improvements matters as much as, or even more than frequent small improvements. A simpler adaptation of this proposal was introduced into our AOS framework [Da Costa *et al.*, 2008; Fialho *et al.*, 2008; Fialho *et al.*, 2009b], and will be considered here too: the credit is set to the maximum fitness improvement over a sliding time-window of size W. This *Credit Assignment* is termed Extreme in the following; more on this in Section 5.2.2.

For all these approaches, in case the mentioned statistics are done over the raw values of the received impact measures, the AOS methods tend to have a problem-dependent behavior, as different problems have different fitness distributions (what alters the range of the fitness improvements received), while also presenting different levels of modality (what also affects the magnitude of the diversity measures). In order to reduce such effect, a normalization over the raw methods can be used, *e.g.*, the credit received by the given operator divided by the highest most recent credit received by all operators. Another yet more robust approach is to discard the raw values, considering their ranks instead. Both normalization and rank-based approaches, which are part of the contributions to AOS proposed in this thesis, will be described in detail and analyzed in Chapter 5.

4.3.3 Whom to assign Credit to?

Another independent issue that has been addressed in different ways in the literature is the choice of the operators that should be credited after the generation of a given offspring. It is unquestionable that the operator used to generate the offspring should be credited; but some authors consider that the operators used to generate its ancestors should also receive a share of its credit, somehow claiming that the generated offspring is as good as it is not only because of its parents and the current operator, but also because of how good were its ancestors and the operators used to generate them.

This is usually done following a kind of bucket brigade algorithm, the credit being assigned with a decay factor for each level of ancestry [Davis, 1989; Julstrom, 1995; Julstrom, 1997]. No clear indication however about the benefits of this approach can be found in the literature to the best of our knowledge; in [Barbosa and Sá, 2000], for example, the use of ancestors (up to 2 levels) was beneficial in some of the continuous

benchmark functions considered, while resulting in worse results on other functions.

Hence, the methods developed during this thesis do not consider ancestry for the *Credit Assignment*: only the operator that has been applied to generate the given offspring is rewarded.

4.3.4 Compass: Aggregating Fitness and Diversity

Besides the fitness improvements, the diversity variation can also be considered somehow for the *Credit Assignment*, specially when tackling multi-modal problems, in order to reward a possible tentative of escaping a local optimum. In an empirical analysis of the AOS schemes within a GA applied to SAT problems (see Section **??**), we have explored Compass [Maturana and Saubion, 2008a], a method that assigns credit to the operators based on an aggregation of both fitness and diversity measures, which works as follows.

A steady-state scheme is used, *i.e.*, the offspring generated after an operator application is instantaneously included into the main population, replacing another individual. Based on this, every time an operator is applied, three impact measures are gathered: (i) population mean diversity variation (ΔD), calculated by means of Hamming distance, (ii) mean fitness or quality variation (ΔQ), and (iii) execution time T, as shown in Fig. 4.2.a. The execution time becomes essential when dealing with complex operators, such as the local search ones used in the Compass original work [Maturana and Saubion, 2008a].

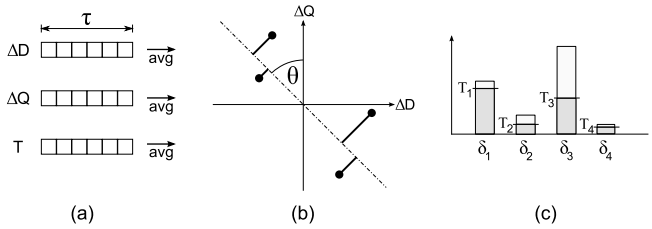


Figure 4.2: Compass credit assignment: Sliding windows of three measures are maintained (a). Average measures of ΔD and ΔQ are plotted and distance of those points are measured according to a plane with a slope of Θ (b). Finally, those distances are divided by the execution time, and the result is the credit assigned to the operator.

Originally, the average of these values over the last τ applications of each operator is displayed in a "diversity *versus* fitness" plot (black dots in Fig. 4.2.b, each dot representing one operator). A user-defined hyper-parameter Θ defines the trade-off between the exploitation (fitness) and the exploration (diversity) criteria, consequently tuning the Exploration versus Exploitation (EvE) balance of the operators selection. In practice, such angle defines the plane according to which perpendicular distances from the dots are measured. Finally, the credit assigned to an operator is this measured perpendicular distance (between the dot representing its performance and the plane defined by Θ), divided by the execution time (Fig. 4.2.c). A complete representation of the Compass *Credit Assignment* technique in the form of a pseudo-algorithm is presented in Algorithm 4.1.

In the Compass original paper [Maturana and Saubion, 2008a], it is combined with the Probability Matching *Operator Selection* scheme (see Section 4.4.1); the resulting

Algorithm 4.1: *Credit Assignment*: Compass (K, Θ)

1: $D_{op} \leftarrow \left(\frac{\text{Average(diversity}_{op})}{\max_{i=1...K} |\text{Average(diversity}_i|)} \right)$ // mean normalized by max

2: $Q_{op} \leftarrow \left(\frac{\text{Average(quality}_{op})}{\max_{i=1...K} |\text{Average(quality}_i|)} \right)$

3: $V_{op} \leftarrow (D_{op}, Q_{op})$ // vector representing *op* in the plot

4: $\alpha_{op} \leftarrow \left| atan\left(\frac{Q_{op}}{D_{op}}\right) - \Theta \right|$ // angle between vector$_{op}$ and plane defined by Θ

5: **return** $\left(\frac{|V_{op}| \cdot cos(\alpha_{op}) - \min_{i=1...K}\{|V_i| \cdot cos(\alpha_i)\}}{\text{Average(exectime}_{op})} \right)$ // distance to plane divided by time

AOS combination is applied to SAT problems, selecting between 6 evolutionary and local search operators. Later on, we established a collaboration with them, in order to combine their sophisticated *Credit Assignment* scheme with our state-of-the-art (by that time) *Operator Selection* mechanism, the Dynamic Multi-Armed Bandit (DMAB), which will be described in Section 5.3.2. A summary of the results achieved by this efficient AOS combination, applied to the same SAT problems, was published in [Maturana *et al.*, 2009a; Maturana *et al.*, 2010a]. These empirical results are revisited in Chapter 6; other examples of schemes using the diversity to calculate the credit to be assigned to an operator after its application are recalled in Section 4.5.2.

4.4 Operator Selection

Based on the credits received from the *Credit Assignment* mechanism after one or more operator applications, most *Operator Selection* schemes maintain an up-to-date empirical quality estimate for each operator, and use it to update their application rates. These probabilities are then used by the underlying algorithm the next time it needs to generate an offspring, and the operator to be applied is selected by a roulette wheel-like process, like in the Probability Matching (PM) [Goldberg, 1990] and Adaptive Pursuit (AP) [Thierens, 2005; Thierens, 2007] methods. Both methods will be detailed in this Section.

Another possibility to *Operator Selection* will be introduced in this work: it is based on the so-called Multi-Armed Bandit framework [Auer *et al.*, 2002], and uses directly the empirical quality estimate gathered by each operator together with an explorative term to deterministically choose amongst the different available operators. This approach is the basis of all *Operator Selection* schemes developed during this thesis work, which will be exhaustively described in Chapter 5.

4.4.1 Probability Matching

Because of its simplicity and reasonable performance, the most widely used to date *Operator Selection* scheme for AOS is the Probability Matching (PM) [Goldberg, 1990]. Although possibly presenting some very slight variations, and sometimes not being explicitly mentioned, PM is used in [Davis, 1989; Julstrom, 1995; Julstrom, 1997;

Lobo and Goldberg, 1997; Barbosa and Sá, 2000; Niehaus and Banzhaf, 2001; Luchian and Gheorghies, 2003; Wong *et al.*, 2003; Whitacre *et al.*, 2006; Maturana and Saubion, 2008a], to mention a few. Its basic idea is that the probability of selecting a given operator is updated proportionally to its known empirical quality with respect to the others. This can be mathematically formalized as follows.

Let K denote the number of available variation operators. PM maintains a probability vector $(p_{i,t})_{i=1,K}$ and an empirical quality estimate for each operator j noted $\hat{q}_{j,t}$. At each time t:

1. The j-th operator is selected with probability $p_{j,t}$, via a roulette-wheel selection scheme.

2. The selected operator is applied, and a credit $r_{j,t}$ is computed after the *Credit Assignment* method at hand;

3. The empirical quality estimate $\hat{q}_{j,t}$ of the j-th operator is then updated to account for this credit received, what is done using an additive relaxation mechanism with adaptation rate α ($0 < \alpha \leq 1$, the memory span decreases as α increases):

$$\hat{q}_{j,t+1} = (1 - \alpha)\, \hat{q}_{j,t} + \alpha \cdot r_{j,t} \tag{4.1}$$

4. And finally, the probabilities of application of each operator, $(p_{i,t})_{i=1,K}$, are updated to be proportional to the their respective empirical quality estimates, $(\hat{q}_{i,t})_{i=1,K}$:

$$p_{i,t} = \frac{\hat{q}_{i,t}}{\sum_{l=1}^{K} \hat{q}_{l,t}} \tag{4.2}$$

By updating the operators probabilities in this way, an operator that performs very badly during a long period of the search will have its application probability decreased to a very low value, or even zero. Such a situation should be avoided, as it would prevent the AOS from using this same operator in case it becomes efficient in a later stage of the search process. For this reason, a minimal selection probability p_{min} is usually enforced. The update rule is then re-defined as follows:

$$p_{i,t+1} = p_{min} + (1 - K * p_{min}) \frac{\hat{q}_{i,t+1}}{\sum_{l=1}^{K} \hat{q}_{l,t+1}} \tag{4.3}$$

A complete representation of the PM *Operator Selection* technique in the form of a pseudo-algorithm is presented in Algorithm 4.2.

Discussion: After Equation 4.3, any ineffective operator (not getting any reward) would have at least a probability p_{min} of being selected. The best operator (getting maximal rewards during some time) would be selected with probability $p_{max} = (1 - (K - 1) * p_{min})$. In practice, however, all mildly relevant operators keep being selected, and this hinders the performance of PM (all the more so as the number of operators increases), as pointed out in [Thierens, 2005].

Algorithm 4.2: *Operator Selection*: Probability Matching (K, p_{min}, α)

for $i = 1$ to K **do**
  $p_i \leftarrow 1.0/K$ // selection probability
  $\hat{q}_i \leftarrow 1.0$ // empirical quality estimate
end for
while NotTerminated **do**
  if one or more operators not applied yet **then**
    $op \leftarrow$ uniformly selected between the operators not applied
  else
    $op \leftarrow$ ProportionalSelectOperator(p) // roulette-wheel
  end if
  Operator op is applied, impacting the search progress somehow
  $r_{op} \leftarrow$ CreditAssignment.GetReward(op)
  $\hat{q}_{op} \leftarrow (1 - \alpha) \cdot \hat{q}_{op} + \alpha \cdot r_{op}$ // relaxation update rule
  for $i = 1$ to K **do**
    $p_i \leftarrow p_{min} + (1 - K \cdot p_{min}) \left(\frac{\hat{q}_i}{\sum_{l=1}^{K} \hat{q}_l} \right)$ // proportional probability update
  end for
end while

4.4.2 Adaptive Pursuit

Originally proposed for learning automata [Thathachar and Sastry, 1985], the Adaptive Pursuit (AP) method was ported to the AOS context [Thierens, 2005] in order to address the above shortcoming of PM. The first three out of the four steps describing PM in Section 4.4.1 are shared by AP: the operators are selected using a roulette-wheel process over their probabilities; after receiving the credit from the operator application, the same relaxation rule is used to update the empirical quality estimates of the operators, as defined in Equation 4.1. The difference is that, in AP, instead of updating the probabilities proportionally to these estimates (see Equation 4.3), a winner-takes-all strategy is employed to push forward very quickly the application probability of the current best operator, noted i_t^*, while consequently decreasing the others, as follows:

$$\begin{cases} i_t^* & = \arg\max_{i=1\ldots K} \{ \hat{q}_{i,t} \} \\ p_{i,t+1} & = \begin{cases} p_{i,t} + \beta \, (p_{max} - p_{i,t}) & if \ i = i_t^* \\ p_{i,t} + \beta \, (p_{min} - p_{i,t}) & \text{otherwise} \end{cases} \end{cases} \tag{4.4}$$

where $\beta \in [0, 1]$ is the learning rate controlling the greediness of the winner-takes-all strategy. The two other hyper-parameters of AP are the same than the ones used in PM: p_{min}, that enforces a minimal level of operators exploration, and the adaptation rate α, which controls the memory span of the *Operator Selection* scheme. A complete representation of the AP technique in the form of a pseudo-algorithm is presented in Algorithm 4.3.

To show the gain brought by the winner-takes-all strategy, in [Thierens, 2005], PM and AP were compared under the light of an artificially generated scenario, choosing between

Algorithm 4.3: *Operator Selection*: Adaptive Pursuit $(K, p_{min}, \alpha, \beta)$

$p_{max} \leftarrow 1 - (K-1) \cdot p_{min}$
for $i = 1$ to K **do**
  $p_i \leftarrow 1.0/K$ // selection probability
  $\hat{q}_i \leftarrow 1.0$ // empirical quality estimate
end for
while NotTerminated **do**
  if one or more operators not applied yet **then**
    $op \leftarrow$ uniformly selected between the operators not applied
  else
    $op \leftarrow$ ProportionalSelectOperator(p) // roulette-wheel
  end if
  Operator op is applied, impacting the search progress somehow
  $r_{op} \leftarrow$ CreditAssignment.GetReward(op)
  $\hat{q}_{op} \leftarrow (1-\alpha) \cdot \hat{q}_{op} + \alpha \cdot r_{op}$ // relaxation update rule
  $op^* \leftarrow \arg\max_{l=1...K}(\hat{q}_l)$
  for $i = 1$ to K **do**
    if $i = op^*$ **then**
      $p_{op^*} \leftarrow (1-\beta) \cdot p_{op^*} + \beta \cdot p_{max}$ // winner-takes-all probability update
    else
      $p_i \leftarrow (1-\beta) \cdot p_i + \beta \cdot p_{min}$
    end if
  end for
end while

5 different artificial operators whose reward distributions were modified every ΔT steps. This artificial scenario, referred to as the Uniform scenario, was also used in our empirical comparisons, and will be described into more detail in Section 5.4.1.

Discussion: Although AP showed a much superior performance than PM in the mentioned artificial scenario, both methods still suffer from two main drawbacks. Firstly, p_{min} defines a minimal level of exploration that is kept fixed during all the search process. Ideally, the surer the *Operator Selection* scheme is about one operator being the best one, the less exploration should be done by it, up to no exploration at all as far as the operator found to be the best remains sufficiently good. A second issue refers to another hyper-parameters, the adaptation rate α: it is also fixed during all the search process, what means that the received credit always has the same fixed weight in the update of the empirical quality estimates of the operators. But, in case there is a long time one operator has not been applied, the assigned credit should have a higher weight, in order to quickly make its empirical quality estimate as up-to-date as possible; conversely, in the case of an operator frequently applied, the reward weight should be smaller, in order to not mess up with its already well-established performance estimate. These issues were part of the main motivations for the proposal of the Dynamic Multi-Armed Bandit (DMAB) and the

Sliding Multi-Armed Bandit (SLMAB) *Operator Selection* techniques; they will be further discussed, respectively, in Sections 5.3.2 and 5.3.3.

4.5 Some Adaptive Operator Selection Combinations

After separately analyzing the AOS components, namely the *Credit Assignment* and the *Operator Selection* schemes, this Section will survey different approaches found in the literature for the AOS as a whole, although most of the cited works were already partially described in the two previous Sections.

The methods discussed here are divided into 5 categories: Section 4.5.1 reviews methods solely based on the fitness value and its derivations; Section 4.5.2 surveys methods that also consider diversity, on its own, or aggregated with fitness; and, due to the number of papers found, for the sake of completeness, methods that use Fuzzy Logic for their operator control are overviewed in Section 4.5.3, although this kind of approach will not be addressed in this thesis. Finally, other approaches that do not match any of the mentioned categories are surveyed in Section 4.5.4, while Section 4.5.5 gives some examples of the use of AOS within EAs other than GA.

4.5.1 Fitness-based Approaches

The seminal AOS method, to the best of our knowledge, was proposed in [Davis, 1989]. Davis' method updates the probability of each operator according to how often its application helped improving the best fitness in the population. A complex decay mechanism is employed to assign credit to the operators that generated the ancestors of the newborn best individual, up to a pre-defined number of generations. Possibly due to the high computational complexity for that time, this technique was not assessed on-line, it was rather used to obtain a non-adaptive time-varying schedule (*i.e.*, a deterministic parameter control scheme, as described in Section 3.3.2) for later use [Tuson and Ross, 1998], which showed to perform better than a GA with fixed operator probabilities.

A similar but much simpler method was proposed in [Julstrom, 1995] and further assessed in [Julstrom, 1997], referred to as Adaptive Operator Probabilities (ADOPP). The most significant differences with respect to [Davis, 1989] are: (i) instead of the best fitness, the median and the 90% quantile of the current fitness distribution are independently tried as reference values for the measure of the fitness improvement; (ii) the rewarding is not based on the raw value of the fitness improvement, but rather on the success rate (1 in case of improvement, 0 otherwise); and (iii) the decay mechanism for the credit assignment to the ancestors is simply done as $(decay^{ancestrylevel} \times credit)$. In [Julstrom, 1995], the ADOPP seems to show good results on a bi-dimensional continuous problem, and on the Traveling Salesman Problem (TSP) problem, although no comparisons with other techniques are presented; while in [Julstrom, 1997], ADOPP is compared with a static strategy (probabilities for each operator fixed at plausible values) on the rectilinear Steiner problem, not being able to achieve better results than it. Note that in both cases, as well as in [Davis, 1989], besides the complex bucket brigade-like *Credit Assignment* scheme,

the *Operator Selection* mechanism used is somehow similar to the PM method, though it is not mentioned.

In [Lobo and Goldberg, 1997], the PM method is used again. An operator is credited whenever improvements over the current best solution are achieved. Better performance is shown with respect to several (static) baseline techniques on the OneMax problem, although the ancestors are not considered on the rewarding scheme, putting into question the use of this complex and expensive (in terms of memory) procedure. In [Barbosa and Sá, 2000], a similar method is tried on the continuous domain: the main difference lies in the aggregation of two fitness improvements, one with respect to the parents, and another in relation to the 90% quantile of the fitness values found in the current population. The use of ancestors up to two levels is tried, credited in the same way as done in [Julstrom, 1997], but no clear evidences are reported to support its use when applied on a set of continuous benchmark problems – indeed, it even degrades the results in some cases.

A very different approach is proposed in [Hatta *et al.*, 1997]: the crossover operator to be applied is chosen according to the *elite degrees* of the individuals selected to be parents. Based on the assumption that an individual which has a large number of recent ancestors with a high fitness value also tends to have a high fitness value, the elite degree of an individual is basically the ratio of the sum of all its "elite ancestors" up to a pre-defined level, divided by the total number of ancestors considered. An individual or ancestor is considered to be an elite member if its fitness is higher than $(\mu + \alpha \times \sigma)$, where μ is the average fitness of the current population, σ the respective standard deviation, and α a user-defined hyper-parameter, referred to as the elite decision factor. Based on this engineered *Credit Assignment* scheme, the *Operator Selection* is deterministically performed as follows: in case the sum of the elite degrees of both parents is higher than another user-defined threshold, a less disruptive operator is applied (the 2-point crossover in this case) in order to try to maintain some of the good building blocks; a disruptive crossover is applied otherwise (the uniform crossover). This work is extended in [Hatta *et al.*, 2001], in which some mutation operators are also considered, and a much more complex scheme is devised to measure the elite degree as a continuous value, instead of the original discrete one. In both works, better results are achieved with respect to the GA independently applying each operator independently, and to the uniform selection between the available operators. Besides, in [Hatta *et al.*, 2001], the scheme implementing the continuous elite degree is shown to improve over the original discrete elite degree, assessed on the NK landscape, the TSP problem, and on a set of continuous benchmark problems.

The Cost Operator Based Rate Adaptation (COBRA) method, devised in [Tuson and Ross, 1998], uses as *Credit Assignment* the average fitness improvements achieved over the parents, divided by the computational cost of evaluating an offspring. No ancestors are considered. The *Operator Selection* is simply done as follows: prior to the experiments, the user defines a set of static probabilities; then, at every adaptation cycle, these probabilities are deterministically assigned to the operators, according to their ranking with respect to the perceived performance measures, the top-ranked operators receiving the highest probabilities. On the *Credit Assignment* side, it is not clear which is the influence of the computational cost, as the evaluation of an offspring is supposed to

have a constant cost, no matter the operator used to generate it. Furthermore, on the *Operator Selection* side, no guidelines are provided on how to define a priori the static set of probabilities; evaluated on the OneMax, Royal Road, Long K-Path and on a deceptive problem, indeed (and not surprisingly), the performance of the COBRA method is found to be dependent on the quality of this user-defined set of probabilities.

A different method, the Probabilistic Rule-driven Adaptive Model (PRAM), proposed in [Ho *et al.*, 1999], uses a sequence of leaning/production phases to adapt the operators rates. During the learning phase, operators are uniformly selected and their performances are estimated based on the fitness improvement of the offspring with respect to its parent. On the following production phase, operators are selected by the PM method, according to the empirical knowledge gathered in the first period. The PRAM method achieves better results than a fixed strategy and a self-adaptive scheme. In [Wong *et al.*, 2003], the PRAM method is used in combination with an external mechanism for diversity maintenance, which gives a higher survival probability to individuals located in a sparsely populated regions of the search space. The resulting method, referred to as APGAIN, consistently achieves better solution quality than several other static evolutionary schemes within a same computational budget, on a set of continuous benchmark problems. However, as pointed out in [Maturana and Saubion, 2008a], around 25% of the generations are devoted to the learning phase, in order to try to accurately follow the changes in the operators performances during the search process, what might severely harm the population and the progress of the search in case disruptive operators are considered.

The Integrated-Adaptive GA (IAGA) [Luchian and Gheorghies, 2003], as its name says, integrates several impact measures to adapt the operators application rates: the frequency of absolute improvements (over the best), simple improvements (over the parents), plateau walks (same fitness than its parents), and worsenings (fitness lower than its parents) achieved by the applications of each operator within a generation. The operators are selected via a PM-like scheme based on the ranks of the operators with respect to the measured frequencies. Besides, the IAGA method also implements an adaptation of some internal parameters of the operators: their description is out of the scope of this Section, we refer the reader to [Luchian and Gheorghies, 2003] for more details. The IAGA method shows to perform much better than the GA using different sets of static probabilities on different instances of the Royal Road problem.

4.5.2 Diversity-based Approaches

Besides the Compass [Maturana and Saubion, 2008a], described in detail in Section 4.3.4, other similar approaches have been proposed in the literature, aggregating the fitness and diversity measures, or using only the diversity as an impact measure after an operator application. Some examples will be briefly recalled in the following.

The Adaptive GA (AGA) proposed in [Srinivas and Patnaik, 1994] is, to the best of our knowledge, the first method proposed for the adaptation of the operators application rates that also take into account the diversity in the decision process, motivated by the difficulty of solving multimodal problems. Its adaptation method can be briefly described as follows. The crossover and mutation rates are varied, for each individual, according

to the difference between the fitness of the best and the fitness of the current individual, divided by the difference between the fitness of the best and the average fitness of the current population. The numerator measures how close to the best individual is the current individual; the fitter the individual is, the less it will be disrupted by the operators. Conversely, the denominator roughly measures the level of convergence of the population; the more converged it is, higher is the variation that should be introduced, in order to possibly escape local optima. The balance between both intensification and diversification (the same as exploitation and exploration, respectively) factors is controlled by a user-defined hyper-parameter for each operator. AGA is shown to significantly outperform a standard GA with fixed operator probabilities on a set of continuous benchmark problems; a much higher gain is achieved in the highly multimodal problems, as expected.

A similar approach, the Diversity-Guided EA (DGEA) [Ursem, 2002], is, as its name says, completely guided by the level of diversity in the population. The main objective is again to avoid premature convergence. A special diversity measure is proposed, referred to as the "distance-to-average-point", which takes into account the size of the search space, the size of the population, and the sum of the differences between the genes of each individual in order to evaluate how converged the population is. Once every generation, the algorithm switches between exploration and exploitation behaviors, based on the assessed diversity level. Intuitively, exploration is performed by the generation of an entire population via the sole use of a mutation operator, while exploitation is done by crossover. Compared to a set of non-adaptive GA schemes on some continuous benchmark problems, the DGEA presents better performance, consistently reducing in around 25% the number of fitness evaluations to attain a given target solution.

The Adaptive Operator Rate Controlled EA (AORCEA) [Giger *et al.*, 2007] is an interesting AOS method, although quite complex. It is very different from the previously mentioned approaches in what concerns the update of the operators application rates. To start with, the criterion to evaluate the impact of an operator application depends on the level of stagnation of the search process, which is calculated based on the frequency distribution of the fitness values of the current population. In case more diversity is needed, applications of operators are evaluated based on how different are the offsprings they generate with respect to their parents (Euclidean distance); the relative fitness improvements (as briefly reviewed in Section 4.3) are used otherwise. The operators are ranked according to how well they perform in average with respect to the chosen criterion during the given adaptation cycle; their application rates are then linearly updated, by taking into account these ranks and the ratio between the level of stagnation and a user-defined hyper-parameter. This hyper-parameter exerts a function analogous to the greediness control β parameter used by AP (described in Section 4.4.2). The AORCEA presents significantly better results when compared to a non-adaptive GA on a set of continuous benchmark functions, and also on a real-world problem, the optimization of the structure of a tubular steel frame for a motorcycle.

Guided by the same motivations than those of the Compass fitness and diversity aggregation method [Maturana and Saubion, 2008a] (see Section 4.3.4), two other *Credit Assignment* mechanisms have been later proposed in [Maturana *et al.*, 2009b; Maturana *et al.*, 2010b], directly inspired by the concepts of Pareto dominance. Con-

sidering the diversification and the intensification as two criteria to be optimized, the first scheme, referred to as *Pareto Dominance* (PD), evaluates the operator according to the number of other operators it dominates, *i.e.*, operators that performed worst in average than the operator under assessment on both objectives. Oppositely, the second scheme, *Pareto Rank* (PR), accounts for the number of operators that dominate the operator under assessment. The main difference between both is that the latter encourages only the use of non-dominated operators, while the former rewards more accurately all the operators that are performing well on average: the PD scheme is thus the best choice, as empirically shown in the cited references. Combined with an external scheme that dynamically changes the set of available operators while solving the problem, the PD *Credit Assignment* scheme with the PM *Operator Selection* mechanism achieves better performance than other adaptive combinations on a set of SAT instances.

4.5.3 Fuzzy-based Approaches

Another kind of approach with several examples found in the literature is the use of Fuzzy Logic Controllers (FLC) to control the selection of operators. Some of these methods will be briefly reviewed now.

The seminal paper on this matter, to the best of our knowledge, is the work by [Lee and Takagi, 1993], in which the operators application rates are deterministically controlled, according to the fuzzy rules, based on population-wise measures: the average, best, and worst fitness values found in the current population. The FLC itself is off-line tuned by another GA at the meta level, according to its performance on the control of the operators of the main GA while solving the well-known set of "5 DeJong functions". The resulting tuned algorithm is later applied to the optimization of another FLC solving the inverted pendulum problem, outperforming a simple static GA in terms of number of evaluations to achieve a given target solution. Although being out of the scope of this thesis, it seems worthy to highlight the several levels of efficient hybridizations between fuzzy and evolutionary techniques that can be found in this work; in summary, a GA is used to optimize an FLC, that controls the operator rates of another GA, which is used to optimize another FLC, that is finally applied to a control problem. Besides, an important motivation for using the kind of human-comprehensive knowledge representation employed by FLCs is that experts can try to incrementally enhance the controller with their own understanding about the problem.

Differently, in [Herrera and Lozano, 2001], an FLC optimized by a meta-GA is used to control the use of 12 different operators by a GA, that is applied to continuous optimization problems. But here the controller is optimized while solving the problem, by means of a separate GA that co-evolves with the GA to be controlled. A gain with respect to non-adaptive schemes is not shown in terms of efficiency, but rather in terms of robustness when applied to continuous benchmark problems with very different levels of difficulty.

Another work using FLC to control the operators application within a GA is presented in [Maturana and Saubion, 2007b; Maturana and Saubion, 2007a; Maturana and Saubion, 2008b]. Similarly to the previously mentioned PRAM method [Ho *et al.*, 1999], the adaptation method is divided into two periods, a learning phase, dur-

ing which the FLC is improved after the empirical knowledge gathered from several trials of all the operators; and a production phase, when the learned rules are actually employed to deterministically select which operator should be applied according to the feedbacks (diversity and quality variations) received from the search. However, around 55% of the generations are spent by the learning phase, what greatly deteriorates the performance of the algorithm, specially if disruptive operators are employed (as also previously discussed for the PRAM method). The presented results do not compare the developed fuzzy-based AOS scheme with other methods from the literature, but rather with different variants of its own, on an instance set of the Quadratic Assignment Problem (QAP).

4.5.4 Other Approaches

Most of the previously cited works use slight variations of the PM method for the adaptation of the rates and further *Operator Selection*, while, as it can be seen, a lot of effort is invested on the many different alternatives mentioned for the *Credit Assignment* part. There is no clear evidence to support the preference for enhancing just one of the modules, but given the difficulty of the task, it might seem relevant to separately investigate both issues. A first step along this line is taken by [Thierens, 2005], which proposes a new mechanism for *Operator Selection*, the Adaptive Pursuit (AP) (see Section 4.4.2), while assessing it on an artificial dynamic scenario, assuming the reward associated to each operator to be known. The reward distributions are modified every ΔT steps, with AP showing a much superior performance than PM. This artificial scenario will be described in detail in Section 5.4.1.

In [Whitacre *et al.*, 2006], attention is given to the *Credit Assignment* scheme again, while the *Operator Selection* is the well-known PM, autonomously selecting between 9 operators. A 10th operator is applied according to a deterministic scheduler in order to enforce some level of diversity. Several alternative impact measures are compared, *e.g.*, the rank of the generated offspring within the current population, and the age of the generated solution; for the latter, the adaptation needs to happen once every many generations (20 in this case). From these impact measures, the main novelty proposed in this work is the use of a *Credit Assignment* mechanism that rewards the production of outlier solutions, which are found out based on statistics over the whole set of generated solutions. This method is shown to be superior than the common Average scheme on a set of continuous benchmark problems.

4.5.5 AOS within Other Evolutionary Algorithms

All the works reviewed in this Section up to now, as well as most of the literature on AOS, are proposed in the context of GAs. However, the concept is general enough to be applied to other EAs (as well as to other meta-heuristics).

For instance, in [Niehaus and Banzhaf, 2001], the PM method is used to select between special operators in the Genetic Programming (GP) framework, based on the success rate of each operator, a successful trial being defined as the generation of an offspring fitter than its parents. The proposed adaptive scheme presents superior performance than the

standard GP using both randomly and empirically defined static application rates, on different problems of symbolic regression and classification.

Some works in the scope of Differential Evolution (DE), yet another EA, can also be found in the literature. For instance, although being referred to as *Self-Adaptive DE*, the SaDE method [Qin *et al.*, 2009] employs indeed the AOS paradigm, with the PM method selecting between DE mutation strategies according to the success rate of each operator. This scheme is combined with another method to dynamically adapt the crossover rate CR and the mutation scaling factor F. The SaDE method outperforms the DE independently applying each of the available mutation strategies, and other previously proposed adaptive and self-adaptive schemes, on a set of continuous benchmark problems.

In a collaboration with the China University of Geosciences [Gong *et al.*, 2010a; Gong *et al.*, 2010b], we have also used the PM method within DE, but this time using the average of the relative fitness improvements as *Credit Assignment*. Along the same lines, a large part of the empirical results presented in Chapter 6 were achieved applying our Rank-based Multi-Armed Bandit AOS mechanisms (which will be described in Chapter 5) to DE on continuous benchmark optimization problems [Fialho and Ros, 2010; Fialho *et al.*, 2010b]. These works will be detailed, respectively, in Sections **??** and 6.6.

4.6 Discussion

Most of the Adaptive Operator Selection (AOS) methods surveyed in this Chapter are employed to select between operators within Genetic Algorithms; a few approaches considering other variants, namely Genetic Programming and Differential Evolution, are also mentioned. Although all these works are in the scope of Evolutionary Algorithms, the *adaptive* paradigm, as reviewed in Section 3.3.2, is indeed very general. In fact, any stochastic algorithm can benefit from this kind of approaches. At a higher level of abstraction, AOS schemes can also be used to select between different algorithms at the hyper level, what is nowadays commonly referred to as *Hyper-Heuristics* (we refer to reader to [Burke *et al.*, 2010] for a recent very comprehensive review on this).

The ideal scenario for the use of the *adaptive* paradigm can be briefly characterized as follows:

1. The algorithm has some choice to be made among different options that directly affect the search process;

2. This choice is supposed to not have only a single best component during the whole search process; instead, different components perform best during different stages of the search;

3. It is possible to have an instantaneous feedback from the search process as a result of the choice.

The first and the third issues are, in fact, requirements to be able to use adaptive methods in general, that indeed quite always hold in the case of stochastic algorithms.

The second issue can be relaxed a bit: even if there is just one unique best option for the given choice, it is usually unknown to the user, and generally problem-dependent; so, the use of an adaptive method can be justified anyway, in order to automatically find the best option. The price to pay in this case is a small loss in terms of performance (the time taken in order to find the best option), which is compensated by the fact that everything is done during an optimization run, while several runs would be needed in order to apply an off-line tuning procedure, as discussed in Section 3.3.2.

In the context of AOS, the input is the feedback received from the search, and the output is the operator to be applied, as depicted in Fig. 4.1. The generality of the method, however, depends on how general is the information used to constantly update it. The methods surveyed in this Chapter use very general information, such as the fitness and the diversity. In some cases, however, one might want to explore some prior knowledge about the characteristics of the problem in order to do a more efficient AOS. A lot of research on this has been done in the very competitive context of SAT problems. An example of a problem-specific AOS method is the recent NCVW (Non-, Clause, and Variable Weighting) [Wei *et al.*, 2008], which uses SAT specific features, the variable and clause weights, in order to choose between three well-known variable selection heuristics. Although losing generality, exploring prior knowledge about the problem can be very beneficial for the search process; indeed, if the motivation is to go for state-of-the-art results, this is very probably the path to be taken in any domain. On the opposite, if the motivation is to have a general method to adapt the operator selection and achieve good performance in very different situations and with different algorithms, such kind of problem-specific knowledge should be avoided.

Finally, as remarked in the bibliographic review of Section 4.5, most of the mentioned works concentrate a lot of effort on just one component of the AOS, usually using a quite common choice for the other one, *e.g.*, lots of methods use a very complex *Credit Assignment* scheme, while implementing the standard PM for *Operator Selection*. In this thesis, we will present different contributions addressing both issues: the bandit-based approaches on the *Operator Selection* side; the Extreme fitness improvement and the *Rank*-based measures over the fitness for *Credit Assignment*. Besides, the Compass [Maturana and Saubion, 2008a] aggregation between fitness and diversity, presented in Section 4.3.4, is also considered after some work done in collaboration with the authors [Maturana *et al.*, 2009a]; the other options for *Credit Assignment* are the common Instantaneous and Average fitness improvement over the parents. All the proposed AOS combinations, presented in Chapter 5, are compared with both PM and AP *Operator Selection* methods, combined with the mentioned *Credit Assignment* schemes.

Part III

Contributions

Chapter 5

Contributions to Adaptive Operator Selection

Contents

In this Chapter, we present our main algorithmic contributions to the Adaptive Operator Selection problem, namely, the extreme and the rank-based approaches for Credit Assignment, the bandit-based techniques for Operator Selection, and the artificial scenarios proposed to their empirical assessment.

5.1 Introduction

As discussed throughout Chapter 4, in essence, the goal of Adaptive Operator Selection (AOS) is to select on the fly the operator maximizing some measure of quality, usually, though not exclusively, reflecting the fitness improvement brought by its application (see, *e.g.*, the Compass method in Section 4.5.2). AOS thus raises two main issues, referred to as the *Operator Selection* and the *Credit Assignment* (Section 4.2). This Chapter describes the contributions developed during this thesis work for each of these issues, as well as for their empirical assessment. A brief summary of these contributions, in a chronological order, is presented as follows.

Starting with the first issue, the *Operator Selection* might be seen as yet another instance of the Exploration versus Exploitation (EvE) dilemma: the operator that is currently known to be the best should be used as much as possible (exploitation), while the other operators should also be tried from time to time (exploration). The exploration needs to be done, on the one hand, because some seemingly poorly-performing operators might just have been unlucky on its recent trials; and on the other hand, due to the dynamics of the evolutionary process, *i.e.*, one of the other operators might eventually become the new best operator at a further moment of the search.

The EvE trade-off has been intensively studied in the context of Game Theory, in the framework of the so-called Multi-Armed Bandit (MAB) [Lai and Robbins, 1985; Auer *et al.*, 2002]. The use of MAB algorithms to solve the EvE dilemma has been investigated in the selection between different algorithm portfolios to solve decision problems [Gagliolo and Schmidhuber, 2008], before being extended to the AOS context in the work presented here. Our preliminary attempt to do so was by directly using the Upper Confidence Bound (UCB) algorithm [Auer *et al.*, 2002], described in more detail in Section 5.3.1. The UCB is an algorithm proposed for the MAB problem that provides asymptotic optimality guarantees with respect to the total cumulated reward. However, these guarantees hold only in a stationary context; some modifications need to be proposed in order to efficiently use the UCB algorithm in the dynamic context of AOS – this is where most of the contributions developed in this thesis are concentrated.

A first proposal, referred to as Dynamic Multi-Armed Bandit (DMAB) [Da Costa *et al.*, 2008], is presented in Section 5.3.2. The DMAB proceeds by coupling the original UCB technique with the Page-Hinkley statistical change-detection test [Hinkley, 1970]: upon detecting a change in the operator quality distribution, the MAB process is restarted from scratch.

Concerning the *Credit Assignment*, most of the AOS combinations found in the literature use some simple statistics over the fitness improvements. Instead of using the common Instantaneous and Average *Credit Assignment* schemes (see Section 5.2.1), we proposed the use of Extreme fitness improvements [Fialho *et al.*, 2008], based on the assumption that outlier high improvements might be even more important than frequent but moderate ones (Section 5.2.2).

The combination of Extreme *Credit Assignment* with DMAB *Operator Selection* referred to as the Ex-DMAB AOS technique, showed to be very efficient, outperforming the baseline approaches on different benchmarking scenarios [Fialho *et al.*, 2008; Fialho *et al.*, 2009a; Maturana *et al.*, 2009a]. However, directly using the raw values of the fitness improvements (*Credit Assignment*) to update the preferences (*Operator Selection*) of the AOS technique showed not to be a very good approach: as different problems have different fitness ranges, this AOS scheme need to have its hyper-parameters tuned for every new problem in order to achieve good performance. For this reason, on the *Credit Assignment* side, we proposed the use of a simple normalization of these raw values [Fialho *et al.*, 2009b] (described in Section 5.2.3).

On the *Operator Selection* side, even with the normalized rewards, the hyper-parameter of the DMAB controlling the change-detection test continued to be very problem-dependent, as the restarting mechanism is directly related to the dynamics of the fitness landscape. This was the main motivation for the proposal of a smoother way to account for dynamic environments in the MAB framework, referred to as Sliding Multi-Armed Bandit (SLMAB) [Fialho *et al.*, 2010a], presented in Section 5.3.3: briefly, it uses a sliding time window to gracefully update the operator quality estimates, discarding ancient events while preserving the information from the recent operator applications. Contrasting with DMAB, the SLMAB does not call upon an external monitoring of the evolution process and involves only 1 hyper-parameter, as the original MAB technique, while DMAB has 2.

The normalized Extreme *Credit Assignment*, however, is still based on the raw values to some extent; thus, the effects of problem-dependency are smoothed, but not eliminated. This is what led us to the proposal of the two last *Credit Assignment* measures, completely based on ranks, the Area-Under-Curve (AUC) and the Sum-of-Ranks (SR) [Fialho *et al.*, 2010c] (Section 5.2.4). Besides the gain in robustness achieved by the use of rank-based measures, using the ranks of the exact fitness values rather than the ranks of the fitness improvements preserves the important invariance of the method with respect to monotonous transformations, being in this case totally comparison-based, as presented in Section 5.2.5. These rank/comparison-based *Credit Assignment* schemes were combined with a simplified version of the UCB, to which we refer to as the Rank-based Multi-Armed Bandit (RMAB) (Section 5.3.4).

By the time this manuscript is being written, the AOS technique constituted by the RMAB *Operator Selection* method with the AUC *Credit Assignment* scheme is our final and recommended proposal in case one wants to implement the AOS paradigm: it achieves state-of-the-art performance while being very robust with respect to their hyper-parameters, as confirmed by the results presented in [Fialho *et al.*, 2010c; Fialho *et al.*, 2010b], which will be detailed in Chapter 6.

Additionally, while developing these AOS combinations, we have also proposed some new artificial scenarios for their empirical assessment. The Boolean and the Outlier scenarios [Da Costa *et al.*, 2008] were introduced to evaluate the AOS schemes in situations involving five artificial operators with different reward distributions than the previously existent Uniform scenario [Thierens, 2005]. The latter is described in Section 5.4.1, while the two newly proposed scenarios are presented in Section 5.4.2. Besides, another family of artificial scenarios was proposed to simulate different situations with respect to the mean and variance of rewards given by two artificial operators, referred to as the Two-Values ($\mathcal{TV}$) benchmarks [Fialho *et al.*, 2010a], presented in Section 5.4.3.

Each of these mentioned contributions is detailed in the following of this Chapter, divided into three categories. Section 5.2 presents the contributions for *Credit Assignment*; while the new methods for *Operator Selection* are described in Section 5.3. At the end of the presentation of each *Operator Selection* scheme, the corresponding AOS combinations are reminded. Finally, Section 5.4 details the newly proposed artificial scenarios for the empirical assessment of the developed AOS combinations. For each Section, the basic or initial approaches are also reminded before the presentation of the contributions, for the sake of self-completeness.

The last contribution of the present thesis will then be a principled and systematic empirical comparison of the proposed AOS methods, compared with one another and with some baseline approaches. Several experiments were done on some different benchmarking scenarios. All the results of these experiments will be detailed in Chapter 6.

5.2 Contributions to Credit Assignment

Credit Assignment is the name given to the scheme that assesses the performance of an operator regarding the progress of the search, which can be measured in different ways, as reviewed in Section 4.3. Starting from the existing Instantaneous or Average of the fitness improvements brought by the application of a given operator (Section 5.2.1), we have proposed the use of Extreme values (Section 5.2.2). In the quest for a higher robustness, a simple normalization over these raw values was firstly proposed (Section 5.2.3), before the development of the most recent and very robust rank-based Area-Under-Curve (AUC) and Sum-of-Ranks (SR) schemes, both described in Section 5.2.4. These latter methods, when using the fitness values instead of the fitness improvements as raw measures of impact, become fully *Comparison*-based, as emphasized in Section 5.2.5.

It is worth remembering that, after discussion in Section 4.3.3, for all the *Credit Assignment* schemes considered (the baseline and the schemes proposed by us), no ancestors are rewarded: the credit is only assigned to the operator that was used to achieve the given fitness improvement. Besides, by convention, all schemes assign a null credit (=0) in case the measured credit is negative.

5.2.1 Basic Credit Assignment Scheme: Fitness Improvements

As discussed in Section 4.3, a *Credit Assignment* scheme is defined by three aspects: (i) how to measure the impact of an operator application; (ii) how to assign credit to the

operator based on this measured impact; and (iii) to which operator the credit should be assigned to.

Concerning the first issue, the *Credit Assignment* schemes proposed in this thesis use as a measure of the impact after an operator application, unless stated otherwise, the fitness improvement brought by the generated offspring over its parent, if a mutation operator is used, or over the best of its parents in the case of crossover. Formally, let $\mathcal{F}$, o and x respectively denote the fitness function, a variation operator, and an element of the current population. The impact of an operator application on the search at time t, *i.e.*, the fitness improvement, is measured as $\delta(t) = (\mathcal{F}(o(x)) - \mathcal{F}(x))$ when the objective is to be maximized, $\delta(t) = (\mathcal{F}(x) - \mathcal{F}(o(x)))$ otherwise. For the description of the *Credit Assignment* schemes throughout this Section, we assume the objective to be maximized; in the case of minimization, the inversion of the credit calculation can be intuitively guessed.

The most common ways of assigning credit based on the impact are used as baseline for comparison, namely: (i) the Instantaneous, which credits the operator according to the fitness improvement received after its most recent application; and (ii) the Average, which assigns as credit the average of the fitness improvements achieved by its last W applications, W being a hyper-parameter that needs to be defined by the user. We refer the reader to Section 4.3.2 for a more extensive discussion including some references for both approaches.

5.2.2 Extreme Fitness Improvement

Our first proposal for the *Credit Assignment* problem, referred to as the Extreme value-based scheme [Fialho *et al.*, 2008], is inspired by the following remark. Let us consider an operator bringing frequent small improvements, and compare it with an operator bringing rare large improvements. The latter operator will hardly be considered if the reward reflects the Average fitness improvement, as the average estimated over a few trials is likely to be 0 (and this becomes even worst in case the Instantaneous values are considered), implying that very few further trials of this operator will take place, although it might be the current best operator.

Hence, as advocated in [Whitacre *et al.*, 2006], attention should be payed to extreme, rather than average, events. Incidentally, the role of extreme events in design has long been acknowledged in numerical engineering (*e.g.*, taking into account rogue waves when dimensioning an oil rig); besides, it receives an ever growing attention in the domain of complex systems, as extreme events govern diffusion-based processes ranging from epidemy propagation to financial markets.

The proposed Extreme value-based *Credit Assignment* mechanism proceeds as follows. When operator o is selected after the *Operator Selection* rule under examination (*e.g.*, PM, AP, or the bandit-based approaches proposed in Section 5.3), o is applied on the current individual x; the fitness of the generated offspring is computed and the improvement achieved over its parents is added to the window of operator o with size W, implementing *FIFO* order. Finally, the credit to be assigned to the operator is set to the maximum value of fitness improvements in this sliding time window.

More formally, let t be the current time step, and t_1 (respectively t_i) denote the time

step where operator o was used for the last time (respectively, the last time before t_{i-1}). If $\delta(t)$ denotes the fitness improvement observed at time t, then the credit to be assigned to operator o is computed as:

$$r_t = \max_{i=1...W} \{\delta(t_i)\} \tag{5.1}$$

Hence, the Extreme value-based mechanism involves a single hyper-parameter, the window size W, as does the previously mentioned Average scheme. This hyper-parameter is meant to reflect the time scale of the process. If too large, the switches between two different situations with respect to operators qualities might be delayed, *i.e.*, operators will be applied after their optimal epoch. If W is too small, operators causing large but infrequent jumps will be ignored, as successful events will probably not be observed at all, or too rapidly forgotten. The Extreme value-based scheme is simply referred to as Extreme in the following.

5.2.3 Normalized Fitness Improvement

Although alleviating the user from the need of selecting which operators should be applied to the problem at hand, and doing so on-line, each of the AOS methods presented in this thesis involves some hyper-parameters that need to be tuned as well. The three *Credit Assignment* schemes previously mentioned, namely, Instantaneous, Average, and Extreme, reward the operators based on simple statistics over the fitness improvements achieved by their application. The use of the raw values of the fitness improvements, however, makes these schemes (and consequently the hyper-parameter tuning of the AOS techniques implementing them) to be very problem-dependent, as follows.

Firstly, different problems have fitness ranges with different variance and at different orders of magnitude; hence, a given AOS setting is efficient just when applied to the problem used during the tuning phase of its hyper-parameters. Additionally, and even more importantly, the fitness variance, as well as the magnitude of the possible rewards received, tend to reduce as the search approaches the optimum, while improvements themselves tend to become more and more scarce; thus, even when the AOS is very carefully tuned for the problem at hand, its behavior cannot be optimal during all the search process.

A proposal to improve the robustness of the mentioned *Credit Assignment* schemes over different problems was to use a simple *Normalization* [Fialho *et al.*, 2009b]: the credit to be assigned to the operator is priorly divided by the maximum credit that would be assigned to any of the operators, according to the *Credit Assignment* scheme under employment (*e.g.*, the mentioned Instantaneous, Average and Extreme schemes). In this way, no matter the moment of the search or the problem that is being tackled, all the rewards are in the real-value interval between 0 and 1, and the current best operator always receives a reward of 1.

The utilization of all these basic *Credit Assignment* schemes described in Sections 5.2.1 to 5.2.3, namely, the absolute or *Normalized* output of Instantaneous, Average, or Extreme schemes, calculated over the fitness improvements, is exemplified in the form of a pseudo-algorithm in Algorithm 5.1, considering a maximization function. It is important to remember that, for all these "basic" schemes, there is one FIFO window of size W for

each of the K operators.

Algorithm 5.1: *Credit Assignment* Schemes over ΔF (op, type, norm, W, K)

```
1: if type = Instantaneous then
2:    reward ← last(wRewards_op)
3: else if type = Average then
4:    reward ← avg(wRewards_op)
5: else if type = Extreme then
6:    reward ← max(wRewards_op)
7: end if
8: if norm then                                                  // normalization
9:    normfactor ← max_{i=1...K}{this.GetReward(i, type, norm=false, W, K)}
10:   reward ← reward/normfactor
11: end if
12: if reward < 0 then
13:   reward ← 0                                                 // rewards ≥ 0
14: end if
15: return reward
```

5.2.4 Rank-based Credit Assignment Schemes

The normalization over the fitness improvement, presented in Section 5.2.3, contribute into reducing the mentioned effects of problem-dependency, but do not eliminate the problem, as it is still based to some extent on the raw values of the fitness improvements. For instance, as the normalization factor depends on the region of the landscape that is currently being explored, the same gain might have different weights in the update of the empirical estimates throughout the search process, and this is likely to lead to problem-dependent hyper-parameter configuration for the AOS schemes.

Inspired by the gain in robustness achieved by GAs when employing rank-based parental selection schemes, *e.g.*, the tournament selection, instead of selection schemes over the raw fitness values, as the fitness-proportional roulette-wheel method (see Section 2.3.3), it seems clear that the use of ranks instead of raw values for the *Credit Assignment* is a way towards robust AOS techniques. Following this path, we have proposed two *Credit Assignment* schemes totally based on ranks, namely, the Area-Under-Curve (AUC) and the Sum-of-Ranks (SR) [Fialho *et al.*, 2010c; Fialho *et al.*, 2010b].

Sliding Window

Besides the fact that ranks are used to assign credit, another major difference is that these newly proposed schemes maintain the gains achieved by all the operators in a single sliding window of size W, still being updated in a *FIFO* way; while in the previously described *Credit Assignment* schemes, there is one separate window for each operator. Each slot in this unique window contains thus the index of the operator that has been applied,

the fitness improvement achieved by its application, and the corresponding ranking with respect to all the other fitness improvements stored in the current window.

The motivation for this is related to the already discussed dynamics of the AOS problem. By the use of just one FIFO window for all operators, the oldest result stored in the window is as old as W operator applications; while in the case of multiple windows, the results of very old applications of a given operator not applied during a long period might still be used for the estimation of its empirical quality, although not reflecting the reality anymore. Besides, as the inclusion of a new value of impact assessment in the window alters the ranking of all the values worse than it (and in case the window contained already W values before this inclusion, the oldest value is deleted from the window; then the ranking of all the results worse than the excluded value are also updated), the application of one operator might also affect the empirical quality estimates of the other operators (in addition to its own), thus automatically handling, to some extent, the dynamics with respect to the performance of the operators.

Decaying Mechanisms for Rank-values

Following, and somehow smoothening the intuition of the Extreme *Credit Assignment* presented in Section 5.2.2, both rank-based approaches that will be presented in the following were tried with the same ranking assignment schemes, which will be now described in turn.

Each position in the window is ranked, with the position R initially receiving a rank-value of $(W - R)$. A decay factor $D \in]0, 1]$ is then applied over these rank-values, so that the top-ranked rewards exert a higher influence on the calculation of the credit assigned to each operator. The final rank value corresponding to each operator application i, which defines its weight in the AUC and SR *Credit Assignment* schemes, is then calculated as:

$$Decay(rank(i)) = D^{R(i)}(W - R(i)) \tag{5.2}$$

The hyper-parameter D, thus, defines how skewed the ranking distribution is. The smaller D, the faster the decay (the more Extreme it is); with $D = 1$ representing the linear decay. We refer to this simply as the *Decay* approach.

Another way of providing a decaying mechanism is the Normalized Discounted Cumulative Gain (NDCG), originally proposed in the context of Information Retrieval (IR) [Järvelin and Kekäläinen, 2000; Burges *et al.*, 2005]. The motivation for using it is very similar: the discovery of highly relevant documents should receive a higher weight than the marginally relevant documents in the evaluation of effectiveness of an IR method. Formally, the original NDCG method assign, to element i (documents in the original context, operator applications in ours), the following rank-value:

$$\text{Original } NDCG(rank(i)) = \frac{2^{R(i)} - 1}{log(1 + i)} \tag{5.3}$$

It seemed to be an interesting method for assigning decayed rank-values to the operator applications, mainly due to the fact that it does the same job, but without needing the definition of any hyper-parameter. In the original context, however, the NDCG measure

does not consider only the ranking (with respect to relevance) of the corresponding document, but also the order (the i in the denominator of Equation 5.3) in which it appears in the documents list: it important for the IR methods to bring the most relevant documents firstly. But in the AOS context, the order in which the top-ranked impact measures are achieved is not important, what matters is how many top-ranked impact measures are brought by each operator, in the time scale limited by the sliding window size W. Therefore, in order to apply the NDCG method in the AOS context, it was re-written as follows:

$$\text{Adapted } NDCG(rank(i)) = \frac{2^{(W-R(i))} - 1}{log(1 + R(i)} \tag{5.4}$$

In practice, however, the NDCG method is rather equivalent to the Decay scheme defined in Equation 5.2 when using $D = 0.4$, as shown in Figure 5.1.

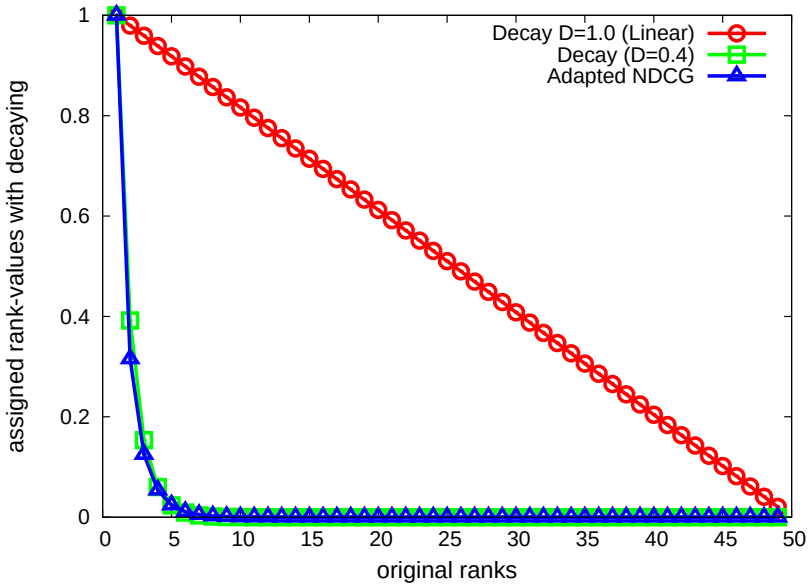


Figure 5.1: Comparison between different decaying mechanisms for $W = 50$.

Anyway, the NDCG and the *Decay* approaches, combined with each of the two rank-based *Credit Assignment* schemes that will be presented in the following, are independently considered in the experiments presented in Chapter 6. In this way, it is possible to verify how much can be gained in terms of performance by trying some different values for D in the *Decay* scheme, with respect to the fixed parameter-less NDCG approach.

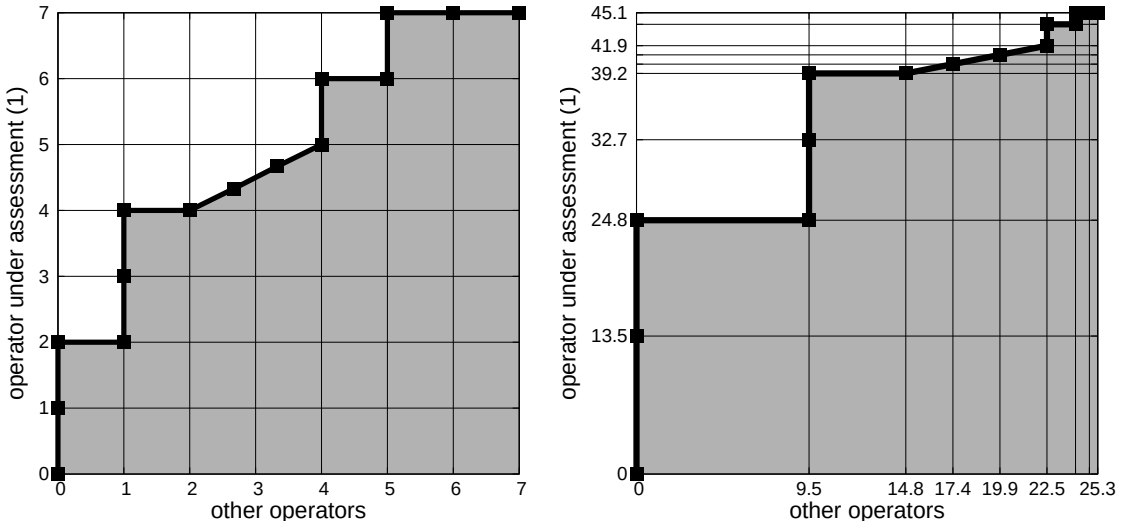
AUC method: Rank-based Area-Under-Curve

The Area-Under-Curve (AUC) *Credit Assignment* method, as its name says, borrows ideas from the *Area Under the ROC Curve*, a criterion originally used in Signal Processing and later adopted in Machine Learning to compare binary classifiers, with the property of being robust with respect to class imbalance [Bradley, 1997]. Originally, the Receiver Operating

Characteristic (ROC) curve depicts how the true positive rate varies with the false positive rate. This indicator is adapted to the rank-based assessment of operators as follows.

Let us consider the list of fitness improvements achieved in a given time window, and let the list be ranked after the raw values of these fitness improvements. The ROC curve associated to a given operator o is drawn by scanning this ordered list, starting from the origin: a vertical segment is drawn when the current offspring has been generated by o, a horizontal segment is drawn otherwise, and a diagonal segment is drawn in case of ties. The credit associated to operator o finally is the area under this curve.

A small example of this procedure is illustrated in Figure 5.2. The ROC curve is the solid line, upper bound of the grey area. The AUC credit is represented by the grey area. While all rank positions have same weight in the example presented in Figure 5.2a (for the sake of clarity), *i.e.*, all horizontal and vertical segments have length 1, it makes sense to give more weight to the top ranked offspring, as previously discussed. A decay factor can be applied in this case, with each segment of the ROC curve being re-scaled according to Equation 5.2, the definition of the hyper-parameter D being required in this case. Figure 5.2b presents the AUC for the same set of rewards, but using decay factor $D = 0.9$.



(a) AUC without decay factor

(b) AUC with decay factor $D = 0.9$

Figure 5.2: Computing the AUC reward associated to operator 1. Four operators are considered; the list of the fitness improvements achieved is sorted; replacing each improvement by the index of the generating operator gives (1 1 3 1 1 2 [3 4 1] 1 2 1 4 4), where [3 4 1] stands for three equal fitness improvement values, resulting in the corresponding slanted segment. In the left figure, no decay factor is applied, each line segment has size 1; in case of decay, the width of the squares decrease leftward and upward, as illustrated in the right figure with $D = 0.9$.

It is important to note that, although using rank-based measures, the range of the credit values provided by AUC is sensitive to the window size: bigger is the window,

exponentially higher will be the credit assigned to the best operator. In order to avoid this situation, we propose again the use of a normalization.

In the preliminary empirical results of the AUC *Credit Assignment* scheme combined with the Rank-based Multi-Armed Bandit (RMAB) (which will be presented in Section 5.3.4) *Operator Selection* mechanism, published in [Fialho *et al.*, 2010c], the normalization was done on a per-axis basis, *i.e.*, in the AUC plot (exemplified in Figure 5.2), the x coordinates (respectively the y ones) are divided by the maximum value to be plotted in x (respectively y). What happens in this case, however, is that different operators will have a different number of rewards being assigned to each axis; consequently, they can be normalized by different values. Consequently, good performance is attained just in situations involving only 2 operators. Later on, in [Fialho *et al.*, 2010b; Fialho and Ros, 2010], we used a much simpler normalization scheme that eliminates such effect. Basically, the AUC credit of a given operator is normalized by the sum of the credits of all operators, so that their sum is equal to 1. This is the current version of the AUC credit, simply referred to as AUC in the remainder of this text, while the preliminary version will be called as AUCv1 for the sake of distinction. An empirical comparison between both versions on the OneMax problem will be presented in Section 6.4.2.

A complete and detailed representation of how to calculate the AUC credit is presented in the form of a pseudo-algorithm in Algorithm 5.2.

Algorithm 5.2: *Credit Assignment*: Rank-based Area-Under-Curve (W, D, op)

$area \leftarrow x \leftarrow y \leftarrow 0$
for rank position $r \leftarrow 1$ to W **do** // loop on window (just one window for all operators)
  $\Delta r \leftarrow D^r(W - r)$ // calculate weight of rank position in the area
  $tiesY \leftarrow CountTiesTargetOp(r)$ // # rewards equal to reward ranked r given by *op*
  $tiesX \leftarrow CountTiesOtherOps(r)$ // # rewards equal to reward ranked r given by others
  if $tiesX + tiesY > 0$ **then** // if ties, proportional diagonal trace
    for rank position $s \leftarrow (r + 1)$ to $(r + tiesX + tiesY)$ **do**
      $\Delta r \leftarrow \Delta r + \left(\frac{D^s(W-s)}{tiesX+tiesY}\right)$ // sum weights of tied ranks, divided by # ties
      $x \leftarrow x + tiesX \cdot \Delta r$
      $area \leftarrow area + y \cdot tiesX \cdot \Delta r$ // area: sum the rectangle below
      $y \leftarrow y + tiesY \cdot \Delta r$
      $area \leftarrow area + 0.5 \cdot \Delta r^2 \cdot tiesX \cdot tiesY$ // area: sum the triangle below slanted line
      $r \leftarrow r + tiesX + tiesY - 1$
    end for
  else if $Op_r == op$ **then** // if *op* generated r, vertical segment
    $y \leftarrow y + \Delta r$
  else // if another operator generated r, horizontal segment
    $x \leftarrow x + \Delta r$
    $area \leftarrow area + (y \cdot \Delta r)$
  end if
end for
return $area / \left(\sum_{i=1}^{K} \text{CreditAssignment.GetReward(W, D, i)}\right)$ // credit = normalized area

SR method: Sum-of-Ranks

The Sum-of-Ranks (SR) is a much simpler method, that credits the operators with the sum of the ranks of the rewards given after its applications, subject to the same decaying mechanism presented in Equation 5.2. This sum is lately normalized by the sum of all the rank-values, so that the sum of the credits assigned to all operators sum up to 1. Formally, the operator i is rewarded at time t as follows:

$$SR_{i,t} = \frac{\sum_{op_r=i} D^r(W-r)}{\sum_{r=1}^{W} D^r(W-r)} \tag{5.5}$$

A complete view of the SR *Credit Assignment* scheme is presented in Algorithm 5.3, which also includes the handling of ties, not represented in Equation 5.5.

Algorithm 5.3: *Credit Assignment*: Sum-of-Ranks (W, D, op)

$sum \leftarrow 0$
for rank position $r \leftarrow 1$ to W **do** // loop on window (just one window for all operators)
  $\Delta r \leftarrow D^r(W-r)$ // calculate weight of rank position in the sum
  $tiesY \leftarrow CountTiesTargetOp(r)$ // # rewards equal to reward ranked r given by *op*
  $tiesX \leftarrow CountTiesOtherOps(r)$ // # rewards equal to reward ranked r given by others.
  if $tiesX + tiesY > 0$ **then** // if ties
    for rank position $s \leftarrow (r+1)$ to $(r + tiesX + tiesY)$ **do**
      $\Delta r \leftarrow \Delta r + \left(\frac{D^s(W-s)}{tiesX+tiesY}\right)$ // sum weights of tied ranks, divided by # ties
    end for
    $sum \leftarrow sum + tiesY \cdot \Delta r$
    $r \leftarrow r + tiesX + tiesY - 1$
  else if $Op_r == op$ **then** // if *op* generated r
    $sum \leftarrow sum + \Delta r$
  end if
end for
return $sum / \left(\sum_{i=1}^{K} \text{CreditAssignment.GetReward(W, D, i)}\right)$ // credit = normalized sum

5.2.5 Comparison-based Credit Assignment Schemes

Coming back to the discussion about the robustness of the *Credit Assignment* schemes in Section 5.2.4, by the use of ranks, both AUC and SR methods can be said to be invariant with respect to linear scaling of the fitness function, *i.e.*, their behavior, when applied on a given fitness function $\mathcal{F}$, is exactly the same than when applied on all the class of fitness functions defined by $(a \cdot \mathcal{F})$, with a real value $a > 0$. Nevertheless, as the raw rewards that are used here are actual values of fitness improvements, some monotonous transformations will indeed modify the ranking of such values, and hence the outcome of the whole algorithm (see some empirical examples on this in Section **??**).

The complete invariance with respect to monotonous transformations can be attained with a very simple modification: the replacement of the fitness improvements by the fitness values of the newborn offspring as a raw impact measure. By doing this, the AUC

and the SR *Credit Assignment* schemes become fully comparison-based, as only sorting some fitness values is required. This means that, in addition to the linear scaling of the fitness functions, they also become invariant to all the family of fitness functions defined by monotonous transformations over the original function.

These comparison-based *Credit Assignment* schemes are referred to as Fitness-based Area-Under-Curve (FAUC) and Fitness-based Sum-of-Ranks (FSR) in the following. As in the original schemes, only the fitnesses of the offspring that improved over their parents are considered, a null reward is assigned otherwise. To date (and to the best of our knowledge), they are the most robust methods for evaluating the operators performance, although being a bit less efficient than the simple rank-based schemes in some cases, as acknowledged in the experimental comparisons presented in Chapter 6.

5.3 Contributions to Operator Selection

Let us turn now to the other component of AOS, the *Operator Selection*, as the process used to select the next operator to be applied, based on the credits received from the *Credit Assignment* scheme under employment during the current search process, as reviewed in Section 4.4. In this thesis, we have proposed and extended the use the Multi-Armed Bandit (MAB) paradigm for *Operator Selection*. Starting with a slightly modified version of the Upper Confidence Bound (UCB), an algorithm for MAB problems, reviewed in Section 5.3.1, we have introduced two different modifications to the UCB, in order to adapt it to account for the dynamics of the AOS problem, namely, the Dynamic Multi-Armed Bandit (DMAB) (Section 5.3.2) and the Sliding Multi-Armed Bandit (SLMAB) (Section 5.3.3). Lately, we have proposed the use of a simplified version of the UCB algorithm, that directly uses the output of the rank/comparison-based *Credit Assignment* schemes, presented in Sections 5.2.4 and 5.2.5, as the empirical quality estimate of each operator, being referred to as the Rank-based Multi-Armed Bandit (RMAB), presented in Section 5.3.4. Each of these contributions to *Operator Selection* will be now described in turn.

5.3.1 Basic Operator Selection Scheme: Multi-Armed Bandit

A very important concept for efficient problem solving within EAs is that of the Exploration versus Exploitation (EvE) balance: as discussed throughout Chapter 2, the EA needs to efficiently *exploit* as much as possible the most promising regions of the search space, while it also needs to *explore* the search space as a whole, in order to have higher chances of finding the true global optimum. In the context of *Operator Selection*, the same EvE problem exists, but at a higher level of abstraction (the algorithm level, not the solution level): the most promising operator needs to be exploited as much as possible, while the other operators also need to be explored from time to time, as the problem of operator selection is dynamic and one of them might become efficient at a further stage of the search. The EvE dilemma has been intensively studied in the context of Game Theory, in the so-called Multi-Armed Bandit (MAB) framework [Lai and Robbins, 1985;

Auer *et al.*, 2002]; based on these works, we have decided to consider the MAB algorithms as possible solutions for the *Operator Selection* problem, as presented in the following.

To start with, the general paradigm for solving Multi-Armed Bandit (MAB) problems can be formalized as follows. A MAB problem involves K arms; the i-th arm is characterized by its (fixed, unknown) reward probability $p_i \in [0, 1]$. At each time step t, the player (game strategy) selects an 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 MAB strategy is measured by 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) best arm, *i.e.*, the arm with maximal probability p^* of getting a reward. Hence, the regret of the strategy after T time steps is:

$$\mathcal{L}(T) = T \cdot p^* - \sum_{t=1}^{T} r_t \tag{5.6}$$

Classically, it is assumed that arms are independent from each other; and that the rewards associated to each arm are independently and identically distributed (*i.i.d.*). Under these assumptions, it can be shown that the optimal regret logarithmically increases with time ($\mathcal{L}(T) = \mathcal{O}(\log(T))$) [Lai and Robbins, 1985]. One of the solutions proposed for the MAB problem, the so-called Upper Confidence Bound (UCB) algorithm [Auer *et al.*, 2002], achieves this optimal regret rate through an Exploration versus Exploitation-based criterion. Formally, the i-th arm is associated to its empirical quality estimate $\hat{q}_i$ (the average of the rewards r_i obtained up to the given time instant); and to a confidence interval, that depends on the number of times n_i the i-th arm has been tried. The UCB algorithm deterministically selects, at each time step, the arm with best upper bound of the confidence interval presented in Equation 5.7.

$$\text{Select} \quad \arg\max_{i=1\ldots K} \left(\hat{q}_{i,t} + \sqrt{\frac{2 \log \sum_k n_{k,t}}{n_{i,t}}} \right) \tag{5.7}$$

$$\text{with} \quad n_{i,t+1} = n_{i,t} + 1 \qquad \text{(number of times used)} \tag{5.8}$$

$$\text{and} \quad \hat{q}_{i,t+1} = \left(\frac{\sum_{j=0}^{t} r_{i,j}}{n_{i,t+1}} \right) \qquad \text{(empirical quality estimate)} \tag{5.9}$$

Clearly, the left term in Equation 5.7 favors the arm with best empirical quality (exploitation), while the right term promotes the trial of the other arms (exploration). The UCB algorithm thus works by choosing mostly the arm that can possibly give the best reward, while giving a chance from time to time to the infrequently tried arms. Its efficiency comes from the fact that, although every arm is selected an infinite number of times, the lapse of time between two selections of an under-optimal arm increases exponentially with respect to the number of time steps; for this reason, the UCB is frequently phrased as *"Be optimistic in front of the Unknown"*.

However, the mentioned optimality of the UCB algorithm with respect to the balance between the Exploration and Exploitation terms is ensured only in the original MAB context, in which rewards are boolean. In the AOS context, the rewards are usually in between some real-value interval, depending on the *Credit Assignment* scheme being employed, what "breaks" this balance; in order to correct it, a *Scaling* factor was introduced by us into the original formula, being represented by the C term in Equation 5.10 [Da Costa *et al.*, 2008]. Besides, in order to avoid storing all rewards received by each operator up to time t, the averaging procedure of the empirical quality estimate presented in Equation 5.9 can be re-written as shown in Equation 5.12.

$$\text{Select} \quad \arg\max_{i=1...K} \left(\hat{q}_{i,t} + \mathbf{C} \cdot \sqrt{\frac{2 \log \sum_k n_{k,t}}{n_{i,t}}} \right) \tag{5.10}$$

$$\text{with} \quad n_{i,t+1} = n_{i,t} + 1 \quad \text{(number of times used)} \tag{5.11}$$

$$\text{and} \quad \hat{q}_{i,t+1} = \left(\frac{(n_{i,t}-1) \cdot \hat{q}_{i,t} + r_{i,t}}{n_{i,t}} \right) \quad \text{(empirical quality estimate)} \tag{5.12}$$

A complete representation of this *Operator Selection* technique is presented in the form of a pseudo-algorithm in Algorithm 5.4.

Algorithm 5.4: *Operator Selection*: Multi-Armed Bandit (K, C)

for $i = 1$ to K **do**
  $n_i \leftarrow 0$ // number of operator trials
  $\hat{q}_i \leftarrow 1.0$ // empirical quality estimate
end for
while NotTerminated **do**
  if one or more operators not applied yet **then**
    $op \leftarrow$ uniformly selected between the operators not applied
  else
    $op \leftarrow \arg\max_{i=1...K} \left(\hat{q}_i + C. \sqrt{\frac{2. \log(\sum_{j=1}^{K} n_j)}{n_i}} \right)$
  end if
  Operator op is applied, impacting the search progress somehow
  $r_{op} \leftarrow$ CreditAssignment.GetReward(op)
  $n_{op} \leftarrow n_{op} + 1$
  $\hat{q}_{op} \leftarrow \left(\frac{(n_{op}-1) * \hat{q}_{op} + r_{op}}{n_{op}} \right)$
end while

It must be noted that the MAB paradigm differs in two aspects 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 AOS: the goal is to continuously learn which operator should be applied while maximizing the fitness improvement in the course of evolution.

Although we acknowledge that MAB is the name given to the problem, for the sake of convenience, in the remainder of this manuscript we refer to the UCB selection strategy with scaling factor C (as shown in Equation 5.10) as the MAB technique. As this technique is the basis of all the *Operator Selection* methods developed during this thesis (presented in the following), it is always used as baseline for empirical comparison, being combined with one of the *Credit Assignment* schemes that use the raw values of fitness improvements to measure the impact of operator applications, namely, the absolute and the normalized versions of the Instantaneous, Average, and Extreme schemes (see Sections 5.2.1 to 5.2.3).

5.3.2 Dynamic Multi-Armed Bandit

As discussed in Section 5.3.1, the MAB algorithm has been designed in order to minimize the regret, *i.e.*, the loss compared to the cumulative reward obtained by the (unknown) optimal strategy (always selecting the best arm/operator) [Auer *et al.*, 2002]. This makes it compulsory to determine *the* best arm (say with reward r): in case the algorithm settles on the second best arm (say with reward r'), it incurs some loss $r - r'$ at each time step, and its regret increases linearly with the number of time steps. In the meanwhile, unpromising arms are tried exponentially less and less; since the reward distribution is assumed to be stationary, the chances of mistaking the best arm for an unpromising one decreases exponentially with the number of trials. The main rationale behind the MAB exploration (trying other arms than the one with best empirical $\hat{q}$) is thus to determine the best arm among the most promising ones.

AOS faces quite different priorities. The main need for exploration comes from the dynamics of the environment: one cannot assume the reward distribution to be stationary, the quality of any operator is bound to vary along evolution. Henceforth, mistaking the best and second best operator has moderate consequences as the loss is small (provided r and r' are sufficiently close) compared to the cost of exploration. The point thus becomes to identify *as fast as possible* a sufficiently good operator.

Note that if the reward distribution is not stationary, the MAB regret cannot but linearly increase with the number of time steps in the worst case. The worst case is when the reward distribution of the supposedly best arm does not change, whereas a previously bad arm covertly becomes the best one. The only way to detect such a worst-case change would be to try all arms sufficiently often, *i.e.*, to define a minimal selection probability, along the same lines as the Probability Matching (PM) *Operator Selection* technique, described in Section 4.4.1. In the evolutionary framework, however, such a worst-case change scenario is unlikely to occur.

On the one hand, the average reward of every operator tends to decrease as evolution goes on (diminishing returns): in the One-Max problem, for instance, the best mutation operator is the 5-bit mutation when the population is far away from the optimum; but the

reward of the 5-bit mutation decreases as the population goes to more fit regions, and at some point the 3-bit mutation operator catches up (more details on this can be found in Section 6.4). This suggests that *when a good operator has been identified, there is no need for exploration as long as this operator remains sufficiently good.*

On the other hand, even without employing a minimal selection probability, and with the lapse of time between two exploration trials increasing exponentially as the search goes on, there is still some exploration being done by the original MAB algorithm – it should thus be able to handle the AOS dynamic scenarios to some extent, although not having been devised to do so. The problem, however, lies in the update rule of the MAB empirical quality estimate $\hat{q}$; the simple averaging formula presented in Equation 5.12 can be re-written as:

$$\hat{q}_{i,t+1} = \hat{q}_{i,t} + \frac{1}{n_{i,t}} \cdot (r_{i,t} - \hat{q}_{i,t}) \tag{5.13}$$

From Equation 5.13, it becomes clear that the weight of the reward received by operator i from the *Credit Assignment* under employment at time t ($r_{i,t}$) is inversely proportional, in the update of the operator empirical quality estimate $\hat{q}_{i,t}$, to the number of times operator i was applied ($n_{i,t}$). Therefore, in case the current best operator has been selected n_i times and its reward falls down by δ, it will need roughly $n_i \cdot \delta/\varepsilon$ trials before recovering an accurate quality estimate of operator i up to precision ε. In other words, the longest is the steady-state of the quality of the best operator, the longer it will take for the MAB process to correct its empirical knowledge in case the situation changes; this *inertia* is what significantly degrades the performance of the original MAB algorithm on the highly dynamic AOS context, as assessed in the experimental comparisons presented in Chapter 6.

The above remarks motivated the proposal of the Dynamic Multi-Armed Bandit (DMAB) approach: the original MAB algorithm is coupled with a statistical test, the Page-Hinkley (PH) change-point detection test [Page, 1954]. Briefly, upon the detection of a change in the reward distribution, the MAB process is restarted from scratch. After describing the PH test, more details about the DMAB are given in the following.

Page-Hinkley Change-Point Detection Test

Given a series of observations $(r_1, \ldots, r_\ell)$, a frequently asked question is whether 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 [Page, 1954], which can be formally described as follows.

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 [Page, 1954]. 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 $|m_t|$ values 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 γ, the PH test is triggered,

i.e., it is considered that the *Change* hypothesis holds. This can be formally written as follows:

$$\begin{cases} \bar{r}_\ell = & \frac{1}{\ell}\sum_{i=1}^{\ell} r_i \\ m_t = & \sum_{\ell=1}^{t}(r_\ell - \bar{r}_\ell + \delta) \\ M_t = & \arg\max_{\ell=1\ldots t}\{|m_\ell|\} \\ PH_t = & M_t - |m_t| \\ \text{Return } & (PH_t > \gamma) \end{cases} \tag{5.14}$$

The PH test involves two hyper-parameters. Parameter γ controls the trade-off between false alarms and unnoticed changes; and parameter δ enforces the robustness of the test when dealing with slowly varying environments. Following early experiments [Da Costa *et al.*, 2008], the latter has been kept fixed to 0.15; while the former is a hyper-parameter that needs to be defined by the user.

MAB + PH = DMAB

The hybridization of the original MAB algorithm (UCB with Scaling factor, as presented in Section 5.3.1) with the PH statistical test, the so-called Dynamic Multi-Armed Bandit (DMAB), thus maintains four indicators for each arm i: from the MAB side, the number $n_{i,t}$ of times this arm has been tried up to time t, and its current (average) empirical quality estimate $\hat{q}_{i,t}$; from the PH test side, there are the average and the maximum deviation measures m_i and M_i. In addition to these indicators, the DMAB also consequently inherits the hyper-parameters of both components, which need to be tuned beforehand by the user, namely, the MAB scaling factor C and the PH change-detection threshold γ (the other PH hyper-parameter, δ, is kept fixed to 0.15, as previously mentioned). A complete and detailed representation of the DMAB behavior, with its indicators and hyper-parameters, is presented in Algorithm 5.5.

Note that the DMAB combination was firstly proposed in another dynamic context by [Hartland *et al.*, 2007], being originally applied in AOS by us in [Da Costa *et al.*, 2008]. In the latter paper, the absolute Instantaneous value of rewards given by some artificial scenarios (described in Sections 5.4.1 and 5.4.2) was used as the *Credit Assignment* scheme, in order to study the *Operator Selection* techniques independently. Being fed by the Extreme value of fitness improvements (and also by the other *Credit Assignment* schemes based on fitness improvements, for the sake of empirical comparison), it was later assessed on some EA binary benchmark problems [Fialho *et al.*, 2008; Fialho *et al.*, 2009a; Fialho *et al.*, 2009b]. It was also evaluated on some SAT instances [Maturana *et al.*, 2009a], within an aggregation of fitness and diversity, the Compass (described in Section 4.3.4), being used as *Credit Assignment*. All these experiments are reminded in detail in Chapter 6.

Algorithm 5.5: *Operator Selection*: Dynamic MAB $(K, C, \gamma, \delta = 0.15)$

for $i = 1$ to K **do**
  $n_i \leftarrow 0$ // number of operator trials (MAB)
  $\hat{q}_i \leftarrow 1.0$ // empirical quality estimate (MAB)
  $m_i \leftarrow M_i \leftarrow 0.0$ // cumulated and max. cumulated difference (PH)
end for
while NotTerminated **do**
  if one or more operators not applied yet **then**
    $op \leftarrow$ uniformly selected between the operators not applied
  else
    $op \leftarrow \arg\max_{i=1...K} \left(\hat{q}_i + C.\sqrt{\frac{2.\log(\sum_{j=1}^{K} n_j)}{n_i}} \right)$
  end if
  Operator *op* is applied, impacting the search progress somehow
  $r_{op} \leftarrow$ CreditAssignment.GetReward(*op*)
  $n_{op} \leftarrow n_{op} + 1$
  $\hat{q}_{op} \leftarrow \left(\frac{(n_{op}-1)*\hat{q}_{op}+r_{op}}{n_{op}} \right)$ // identical to the MAB algorithm up to here
  $m_{op} \leftarrow m_{op} + (r_{op} - \hat{q}_{op} + \delta)$ // then the PH change-detection test is performed
  if $|m_{op}| > M_{op}$ **then**
    $M_{op} \leftarrow |m_{op}|$
  else if $M_{op} - |m_{op}| > \gamma$ **then**
    Restart MAB and PH variables $(n, \hat{q}, m, M)$
  end if
end while

5.3.3 Sliding Multi-Armed Bandit

Although showing to be very efficient, the DMAB *Operator Selection* technique presents two main weaknesses. On the one hand, the change-point detection test is triggered only in the case of an *abrupt* change, whereas the reward of an operator usually decreases gradually: this makes it very difficult to calibrate this test (namely, its change-detection threshold γ). On the other hand, upon triggering the test, the whole memory of the MAB process is lost, and the exploration of the operators must start anew.

These remarks motivated the introduction of a new bandit-based *Operator Selection* technique, referred to as the Sliding Multi-Armed Bandit (SLMAB) [Fialho *et al.*, 2010a]. The underlying idea of this method is to be able to gracefully follow the dynamics of the AOS scenario, without needing the very sensitive and somehow controversial (although efficient when correctly tuned) restart mechanism employed by DMAB.

Several heuristics have been proposed to update statistical estimates in a non-stationary context. The most natural heuristic is the so-called relaxation update rule, used by the PM and AP schemes (presented in Sections 4.4.1 and 4.4.2, respectively), where the weight of the instant reward r_t on the update of the empirical quality estimate

is defined by some constant learning rate α ($0 < \alpha \leq 1$):

$$\hat{q}_{i,t+1} = (1 - \alpha) \cdot \hat{q}_{i,t} + \alpha \cdot r_{i,t} \tag{5.15}$$

The difficulties with the above rule are that, besides introducing the extra hyper-parameter α to be tuned by the user, it defines a constant weight for the instant reward $r_{i,t}$, regardless of how frequently the i-th operator has been applied in the last time steps. In the AOS framework, however, different operators are applied with different frequencies; if an operator has not been applied for a long time, the weight of the instant reward it received should be higher, everything else being equal, in order to enable a more rapid adjustment of its $\hat{q}_{i,t}$.

The update rule must thus take into account the number of time steps elapsed since the previous time step t_i in which the i-th operator has been applied. Finally, in order to preserve the MAB trade-off between exploration and exploitation, one must also maintain the $n_{i,t}$ counters reflecting the frequency of application of operators up to time step t.

Considering a window of size W, the update of the sliding exploitation and exploration terms (respectively, $\hat{q}$ and n), which is performed every time operator i is applied, is defined as:

$$\begin{cases} \hat{q}_{i,t+1} & = \hat{q}_{i,t} \cdot \frac{W}{W+(t-t_i)} + r_{i,t} \cdot \frac{1}{n_{i,t}+1} \\ n_{i,t+1} & = n_{i,t} \cdot \left(\frac{W}{W+(t-t_i)} + \frac{1}{n_{i,t}+1} \right) \end{cases} \tag{5.16}$$

The above update rule is designed in such a way that, if an operator is applied with frequency W/n_t, then n_t is constant. The rationale for this update scheme can be explained as follows.

If an operator is performing well and is almost always applied, counter $n_{i,t}$ rapidly increases up to W and sticks to this value, while its empirical quality estimate $\hat{q}_{i,t}$ accurately reflects the reward expectation for the current stage of the search. The main difference compared to the MAB and DMAB settings is that $n_{i,t}$ is upper bounded by W. Equivalently, the inertia of the reward estimate is bounded: the weight of the instant reward cannot be less than $1/W$.

Oppositely, if an operator is rarely applied, its $\hat{q}_{i,t}$ can be seen as an outdated, hence probably optimistic, estimation of the actual reward expectation (assuming that the operator reward decreases on average as evolution goes on). On the other hand, the fact that the operator is rarely applied means that its empirical quality estimate is lower than that of the current best operator. With the averaging update rule employed by the original MAB scheme, presented in Equation 5.12, it would take a long time to correct the empirical quality estimate of this operator in case it had become the new best one. With the sliding update rule, however, this outdated estimation of the operator quality is more efficiently corrected: if the operator has not been tried in the previous W time steps, $n_{i,t}$ is low, consequently the weight given to the instant reward is high in the update formula, thus rapidly shifting the empirical quality estimate towards its actual value.

Besides the scaling factor C, that is needed by all bandit-based *Operator Selection* mechanisms, the other hyper-parameter that needs to be defined in SLMAB is the window

size W, used in the proposed window-based relaxation update mechanism. But, as most *Credit Assignment* schemes found in the literature, including the schemes proposed in this thesis, rely on windowing the operator reward distribution, this latter hyper-parameter can be said to be parametrized "for free" (by using the same window size W employed by the *Credit Assignment* scheme being used). A complete presentation of the SLMAB in the form of a pseudo-algorithm is presented in Algorithm 5.6.

Algorithm 5.6: *Operator Selection*: Sliding Multi-Armed Bandit (K, C, W)

$times \leftarrow 0$ // number of total time steps
for $i = 1$ to K **do**
  $n_i \leftarrow 0$ // number of operator trials
  $\hat{q}_i \leftarrow 1.0$ // empirical quality estimate
  $last_i \leftarrow 0$ // last time it was applied
end for
while NotTerminated **do**
  if one or more operators not applied yet **then**
    $op \leftarrow$ uniformly selected between the operators not applied
  else
    $op \leftarrow \arg\max_i \left(\hat{q}_i + C \cdot \sqrt{\frac{2 \cdot \log(\sum_{j=1}^{K} n_j)}{n_i}} \right)$
  end if
  Operator op is applied, impacting the search progress somehow
  $r_{op} \leftarrow$ CreditAssignment.GetReward(op)
  $\hat{q}_{op} \leftarrow \hat{q}_{op} \cdot \left(\frac{W}{W + (times - last_{op})} \right) + r_{op} \cdot \left(\frac{1}{n_{op}+1} \right)$
  $n_{op} \leftarrow n_{op} \cdot \left(\frac{W}{W + (times - last_{op})} + \frac{1}{n_{op}+1} \right)$
  $last_{op} \leftarrow times$
  $times \leftarrow times + 1$
end while

The SLMAB *Operator Selection* technique, combined with the absolute Instantaneous, Average and Extreme *Credit Assignment* schemes, was assessed on some artificial scenarios, described in Section 5.4, being also used to select between some mutation and crossover operators within a real EA applied to the Royal Road benchmark problem [Fialho *et al.*, 2010a]. All these experiments are detailed in Chapter 6.

5.3.4 Rank-based Multi-Armed Bandit

The main criticism with respect to the bandit-based AOS schemes described up to now is related to the high sensitivity of their hyper-parameters, what was an important motivation factor for most of the further developments presented throughout Sections 5.2 and 5.3. From the *Operator Selection* point-of-view, the SLMAB represented a further step towards more robust schemes, by eliminating one of the two very sensitive hyper-parameters present in the DMAB approach (the Page-Hinkley change-detection threshold

γ), while still being able to achieve equivalent performance in following the AOS dynamics on some artificial benchmark functions [Fialho *et al.*, 2010a], as presented in Chapter 6. But, even for the SLMAB, there is still the need to tune the scaling factor C, which is indeed a very sensitive hyper-parameter common to all bandit-based *Operator Selection* approaches previously presented. The difficulty for tuning this parameter comes in fact from the *Credit Assignment* scheme being employed, as follows.

By the time the DMAB and SLMAB techniques were proposed, the *Credit Assignment* schemes under consideration were the basic Instantaneous, Average (Section 5.2.1), and Extreme ones (Section 5.2.2), as well as their *Normalized* versions (Section 5.2.3). All these schemes assign credit based directly on some statistics over the raw values of the fitness improvements. Whenever these raw values are used, the tuning of the AOS hyper-parameters tends to be highly problem-dependent, as the range of fitness values varies widely from one problem to another, as well as in the course of an optimization run (we refer the reader to Section 5.2.4 for a more extensive discussion on this issue). Hence, the scaling factor C, which has as original role to tune the balance between the exploitation and exploration terms of the UCB formula (Equation 5.10), also needs to play a radically different role, that of accounting for the scale of the rewards received. This double role is the reason why C shows to be such a sensitive hyper-parameter.

These issues motivated the proposal of the rank-based and further comparison-based *Credit Assignment* schemes, presented in Sections 5.2.4 and 5.2.5, respectively. However, the direct combination of these schemes with the previously described bandit-based *Operator Selection* techniques showed a rather poor performance after some preliminary experiments. The reason for this is that the AUC and the SR indicators already provide, on the *Credit Assignment* side, an empirical statistics over the last W offspring generated; while the MAB techniques do another aggregation of rewards in the *Operator Selection* side (the $\hat{q}$ in Equation 5.10) – the two layers of statistics were somehow diluting the interesting characteristics of the proposed performance measurements. Therefore, as the outputs of the AUC and SR indicators already reflect accurate and up-to-date performance measures of one operator with respect to all the others, they can be used directly as the exploitation term in the MAB formula, *i.e.*, $\hat{q}_{i,t} = AUC_{i,t}$ or $SR_{i,t}$, depending on the scheme being used.

This simple adaptation of the MAB scheme brings another very important benefit for *Operator Selection*. As the mentioned rank/comparison-based *Credit Assignment* schemes maintain just one sliding window for the rewards received by all operators, the inclusion of a new reward in the sliding window, achieved by a given operator, affects the quality estimates of all the other operators. Consequently, the AOS dynamics are already handled on the *Credit Assignment* side in a transparent way, without needing an external observer, as the change-detection test in the DMAB technique, or a relaxation update rule, as the one employed in the SLMAB scheme.

Finally, in order to ensure a minimal level of exploration, the MAB term n is modified to reflect the number of times each operator appear in the sliding window. A complete representation of this simplified version of the MAB algorithm, specially adapted to be used with the rank/comparison-based *Credit Assignment* schemes, thus referred to as the Rank-based Multi-Armed Bandit (RMAB), is presented in the form of a pseudo-algorithm

in Algorithm 5.7.

Algorithm 5.7: *Operator Selection*: Rank-based Multi-Armed Bandit (K, C)

1: **for** $i = 1$ to K **do**
2: $n_i \leftarrow 0$ // number of times operator i appears in the current credit window
3: $\hat{q}_i \leftarrow 1.0$ // empirical quality estimate = normalized CreditAssignment output
4: **end for**
5: **while** NotTerminated **do**
6: **if** one or more operators not applied yet **then**
7: $op \leftarrow$ uniformly selected between the operators not applied
8: **else**
9: $op \leftarrow \arg\max_i \left(\hat{q}_i + C \cdot \sqrt{\frac{2 \cdot \log(\sum_{j=1}^{K} n_j)}{n_i}} \right)$
10: **end if**
11: Operator op is applied, impacting the search progress somehow
12: **for** $i = 1$ to K **do** // the application of one operator might affect the others
13: $\hat{q}_i \leftarrow$ CreditAssignment.GetReward(i)
14: $n_i \leftarrow$ CreditAssignment.GetTimes(creditWindow, i)
15: **end for**
16: **end while**

In order to evaluate the efficiency and robustness of the AOS techniques derived from the combination of the RMAB *Operator Selection* scheme with the rank/comparison-based *Credit Assignment* schemes, they were firstly assessed within a GA applied to the OneMax problem and to other three fitness functions defined by monotonous transformations over this problem [Fialho *et al.*, 2010c]. Later on, their performances were also evaluated on a set of 24 single-objective continuous problems [Fialho *et al.*, 2010b; Fialho and Ros, 2010], selecting between different mutation strategies within a DE algorithm. As expected, these combinations showed to be very robust with respect to their hyper-parameters, namely, the scaling factor C, the decay factor D, and the window size W; while achieving the same level of state-of-the-art performance of the previously presented (efficient but very problem-dependent) approaches. The experimental results on the OneMax and on the continuous problem will be detailed, respectively, in Sections 6.4 and 6.6.

By the time this manuscript is being written, the use of RMAB as *Operator Selection*, with the Area-Under-Curve (AUC) *Credit Assignment* scheme (see Section 5.2.5), is our final recommended choice in case one wants to use the AOS paradigm on his own optimization algorithm/problem. The reason for this choice will be extensively discussed and justified in Section 6.8, based on the evidences brought by the comprehensive empirical analysis that will be presented in Chapter 6.

5.4 Contributions to Empirical Assessment

While developing the AOS schemes presented in this Chapter, some artificial benchmark problems were proposed to analyze different aspects of their behavior in a controlled environment. All these artificial scenarios, that will be now described in turn, were used in part of the empirical analysis of the AOS schemes, which will be presented in Section 6.3.

Based on the Uniform artificial scenario, proposed in [Thierens, 2005] and described in Section 5.4.1 for the sake of self-containedness, we have introduced two other artificial scenarios. Referred to as the Boolean and the Outlier scenarios, presented in Section 5.4.2, they involve the same switches between five operators, but with rewards coming from different distributions and with different probabilities.

More recently, we have extended this set of artificial benchmarks by proposing a new family of problems, referred to as the Two-Values ($\mathcal{TV}$) benchmarks [Fialho *et al.*, 2010a], which can be used to simulate different situations with respect to the mean and variance of rewards given by two artificial operators. The $\mathcal{TV}$ benchmarks are described into detail in Section 5.4.3.

5.4.1 Base Artificial Scenario: Uniform

The Uniform artificial benchmark, proposed in [Thierens, 2005], involves a set of 5 operators, in which the reward distribution associated to each operator is constant during an epoch (ΔT times steps). During every epoch, the operator reward is uniformly drawn from an interval, as follows: $\{4, 6\}$ for the current best operator, $\{3, 5\}$ for the second best, $\{2, 4\}$ for the third, $\{1, 3\}$ for the fourth, and $\{0, 2\}$ for the worst operator.

The reward distributions associated to all operators are permuted at the end of every epoch, using pre-defined permutations to decrease the experimental noise, defined in this work as follows: $41203 \mapsto 01234 \mapsto 24301 \mapsto 12043 \mapsto 41230 \mapsto 31420 \mapsto 04213 \mapsto 23104 \mapsto 14302 \mapsto 40213$. More precisely, the best operator in the first epoch is the op_4, which becomes the worst one in the second epoch. The best operator in the second epoch is op_0, which was the fourth one in the first epoch.

The AOS ability to match the dynamics of evolution is assessed by varying the length of the epoch, *e.g.*, set to $\Delta T = 50$ for fast dynamics and $\Delta T = 200$ for slow ones. The performance associated to an AOS scheme is the cumulative reward obtained during this sequence of 10 epochs. As the reward expectation of the best operator is 5, the maximal cumulative reward is 2,500 in the fast case ($5 \times 10 \times 50$) and 10,000 in the slower one ($5 \times 10 \times 200$).

5.4.2 Boolean and Outlier Scenarios

Within the Uniform benchmark, an operator always gets a reward that is positive, while also being informative, *i.e.*, it indicates (to some extent) which is the best operator, possibly mistaking just with the second best, as the intervals of their reward distributions overlap. In a real evolutionary context, however, the AOS task might be much more chal-

lenging. For instance, the probability of getting some useful information about the quality of the operator might be smaller than that, up to the situation in which no information whatsoever is provided to the AOS (specially true when the search is getting closer to the optimum); or, even if some rewards are frequently provided, the information gathered from them might not be so useful in order to efficiently differ between the available operators. Based on these two difficulties, we proposed [Da Costa *et al.*, 2008] and further used [Fialho *et al.*, 2010a] two variants of the Uniform benchmark, referred to as the Boolean and Outlier scenarios.

In the Boolean scenario, the best operator gets a reward of 10 with probability 50% (and 0 otherwise); the second best gets the same reward of 10 but with probability 40%, and 0 otherwise, and so forth, until the worst operator, getting a reward of 10 with probability 10% and 0 otherwise. In this scenario, the difference between the operators is the probability of getting a non-null reward; the reward takes the same value in all cases. In particular, the best operator has the same reward expectation than in the Uniform scenario, though with a much higher variance.

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

Summarizing, thus, the probability of getting some information is high (for the best operator) in the Boolean benchmark, while being low (for all operators) in the Outlier benchmark. But the Boolean scenario typically does not provide useful information (all rewards have the same values, only the probabilities differ), while the Outlier scenario involves very informative but rare rewards. Both use the same sequence of switches between reward distributions after every epoch, and the reward expectation for each operator is also the same as in the Uniform case.

5.4.3 Two-Value Scenarios

As discussed in the previous Section, any AOS is usually provided with some (more or less) informative results (the reward amount, everything else being equal); and it is more or less likely to be provided with any information at all. Typically, the MAB process is well equipped to deal with Boolean-like settings, where operators (arms) get the same reward in case of success and only the probability of success differs.

Along these lines, a framework for AOS benchmarks, referred to as Two-Values ($\mathcal{TV}$) benchmarks, was proposed in [Fialho *et al.*, 2010a], enabling a more precise control of the two issues previously discussed. Briefly, every operator is assumed to get one out of two reward values, a small value noted r and a large one noted R. Within these two parameters, it is possible to control the informativeness of the reward distribution, defined by the ratio R/r; while the third parameter, p, defines the probability of getting reward R; and $(1-p)$ is the probability of getting r. Needless to say, the mean and the variance of the rewards received are also intrinsically managed by these parameters.

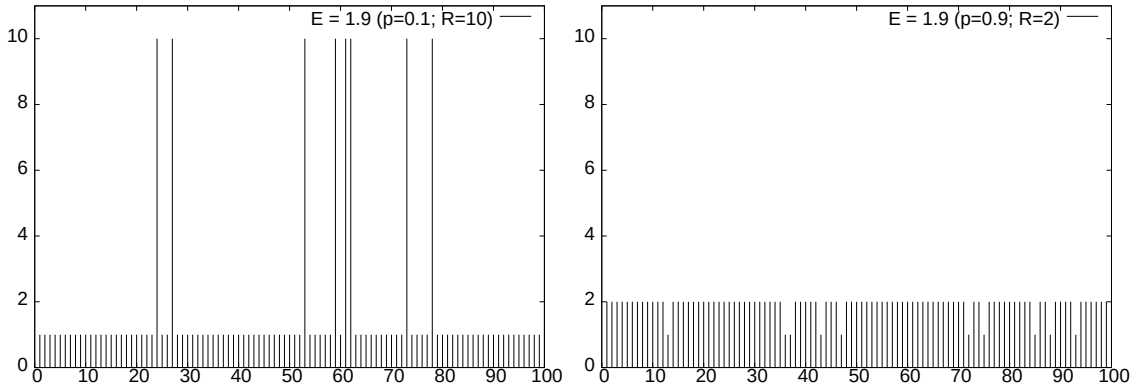
Formally, the reward distribution specified from the triple (p, r, R) is defined as:

$$\begin{cases} \mathcal{TV}(p,r,R) = & R \text{ with probability } p \\ & r \text{ with probability } 1-p \end{cases}$$

with expectation and variance respectively noted $\mathbb{E}(p, r, R)$ and $V(p, r, R)$:

$$\begin{aligned} \mathbb{E}(p,r,R) &= p \cdot R + (1-p) \cdot r \\ V(p,r,R) &= p \cdot (1-p) \cdot (R-r)^2 \end{aligned}$$

It is clear that only the ratio R/r impacts the results; thus, in the remainder of this manuscript, r will be set to 1 and omitted in the notations for the sake of simplicity – a reward distribution will be noted $\mathcal{TV}(p, R)$ instead of $\mathcal{TV}(p, 1, R)$. Furthermore, a scenario involving $\mathcal{TV}(p_1, R_1)$ and $\mathcal{TV}(p_2, R_2)$ will be denoted by $\mathcal{ARTV}(p_1, R_1, p_2, R_2)$. The respective roles of p and R are exemplified in Figure 5.3, displaying two samples of size 100 of distributions with the same expectation and high versus low variance.



(a) $\mathcal{TV}(0.1, 10)$, $V(.1, 10) = 7.29$ (b) $\mathcal{TV}(0.9, 2)$, $V(.9, 2) = .09$

Figure 5.3: Two samples drawn from two $\mathcal{TV}$ distributions with same expectation $\mathbb{E}(.1, 10) = \mathbb{E}(.9, .2) = 1.9$; the distribution on the left picture presenting a much higher variance ($V(.1, 10) = 7.29$) then the one on the right ($V(.9, 2) = .09$).

Such a general framework will help us into analyzing very different aspects of the behavior of the AOS schemes proposed. The main question will always be how agile a given AOS combination is into switching between two different situations, what will be analyzed under the light of different variants of this scenario in Section 6.4. But other low level details could also be analyzed, *e.g.*, does the AOS (or the *Credit Assignment* scheme it implements) tend to favor operators with a high variance instead of the ones with a low variance [Fialho *et al.*, 2010a].

In the same way than for the previously presented scenarios, the exchanges between the reward distributions, done after each epoch of ΔT time steps, obey a fixed sequence in order to decrease the experimental noise, as follows: $01 \mapsto 01 \mapsto 10 \mapsto 01 \mapsto 10 \mapsto 10 \mapsto 10 \mapsto 01 \mapsto 01 \mapsto 10$.

5.5 Discussion

In order to be well-accepted by the EA community, the main requirement of a good AOS technique is, of course, to be able to efficiently automate the control of the operators to be applied, in a dynamic way, during the search process. But it also needs to be (i) easily implementable and, even more importantly, to be (ii) computationally cheap.

The first issue can be alleviated by making available implementations of the proposed schemes (*e.g.*, as open source libraries). The algorithms implemented in this work were coded in *ANSI-C++*; the *Evolving Objects* (EO) library[1] [Keijzer *et al.*, 2002] was used to assist the implementation of the underlying EAs, controlled by the proposed AOS methods. All this experimental framework is freely available under request by email. And indeed, it was already requested by some researchers, and we acknowledge its recent use by one of them [Verel *et al.*, 2010] on the proposal of a new method.

A lot of care should be taken, however, with the second issue: the more computationally expensive the AOS technique, the smaller the margin of possible benefits it can bring to the underlying algorithm which operators are being controlled by it, the benefits being usually measured in terms of computational time or effort to achieve a given solution. Indeed, in some domains, *e.g.*, in numerical engineering, the fitness evaluation is very expensive, what makes the AOS computational cost negligible. But in combinatorial problems, such as the Boolean Satisfiability problems (see Section 6.5.2), the cost of evaluating the fitness of a given solution is almost zero. In this latter case, it might become impracticable to consult and update the AOS technique every time an operator is applied: a trade-off should thus be found between the AOS granularity (how frequent it is consulted and updated, *e.g.*, once every generation instead of once every application) and the AOS dynamics accuracy (the more frequent it is updated, the more reliable will be its estimation with respect to the operators empirical performance).

A third issue worth discussing concerns a very common critic that prevents people from using AOS schemes on their own algorithms and problems: although being proposed to automate some user choices, these schemes have their own (hyper-)parameters that need to be tuned in order to achieve acceptable performance, *e.g.*, the scaling factor C and the window size W, respectively, common to the all *Operator Selection* schemes and *Credit Assignment* mechanisms we have proposed. What should be argued in this case is that the AOS schemes automatically take care of many shallow parameters (*i.e.*, which operators should be applied, and at which rate, besides the main fact that they control the rates in a dynamic way, during the search process), while involving only a few general hyper-parameters (2 or 3, depending on the AOS combination being implemented). Moreover, these hyper-parameters usually have a much clearer meaning than the original ones, and are thus more easily understood by humans, while hopefully being less sensitive to different settings.

A very common approach to solve this hyper-parameter setting issue is to use off-line parameter tuning methods to automatically set them. Indeed, this has been done for all the experiments that will be presented in Chapter 6, in order to promote a fair empirical

[1] EO: C++ library for coding EAs, freely available at http://eodev.sourceforge.net/ as of today.

comparison between the proposed and the baseline techniques. But off-line tuning is an expensive procedure, as discussed earlier in Section 3.3.2; so, ideally, a good AOS technique should also be robust with respect to its hyper-parameters, *i.e.*, whenever a new problem needs to be solved, the tuning of the AOS hyper-parameters should be required as rarely as possible.

Although a good level of efficiency with respect to the AOS dynamics was attained very early in this thesis work, with the proposal of the Ex-DMAB technique (combining the Extreme *Credit Assignment*, presented in Section 5.2.2, with the DMAB *Operator Selection*, described in Section 5.3.2), the hyper-parameters of such AOS combination showed to be very sensitive, a good performance being shown just in case the hyper-parameters were tuned for every new problem. All the further developments proposed in this thesis, for both *Credit Assignment* and *Operator Selection* issues, aimed at smoothening this effect by creating more robust (*i.e.*, less problem-dependent) techniques, while maintaining the same level of performance.

A big progress was achieved on this direction by the recent proposal of the RMAB for *Operator Selection* (Section 5.3.4), combined with any of the rank-based *Credit Assignment* schemes described in Sections 5.2.4 and 5.2.5. The simple fact that rewards are assigned based on ranks instead of raw values of fitness improvements already provides a much higher robustness to the AOS technique implementing it, guaranteeing invariance with respect to any linear scaling of the original fitness function. Furthermore, the use of ranks over the fitness values of the generated offspring, instead of ranks over the fitness improvements, provides to the AOS technique the characteristic of being fully comparison-based, *i.e.*, invariant with respect to all monotonous transformations over the same original fitness function. This robustness gain was confirmed by performing independent off-line tuning procedures over the AOS combination constituted by the Fitness-based Area-Under-Curve (FAUC) as *Credit Assignment* with the RMAB as *Operator Selection* over very different problems – the same or very similar hyper-parameter settings were found to be the best, while state-of-the-art (or very close to that) performance was achieved, as reviewed in the empirical comparisons presented in Chapter 6.

The proposed rank-based AOS techniques are efficient and robust, but it is important to note that their current implementations can not achieve good performance in problems with high multi-modality (*i.e.*, several local optima), because the only action being rewarded is the progress with respect to the fitness. As exemplified in Section 4.5.2, in order to efficiently tackle multi-modal problems, the maintenance of some diversity in the population should also be rewarded somehow. Further developments should be done in this direction in the near future by, *e.g.*, trying to provide the same level of robustness and invariance properties to the *Credit Assignment* schemes proposed in [Maturana *et al.*, 2010b], which aggregate both fitness and diversity measures to evaluate the operator performance. A different alternative, in the case of multi-modal problems, would be to let the proposed AOS techniques as they are, *i.e.*, rewarding just exploitation (fitness), but implementing efficient convergence detection mechanisms in the underlying EAs, in order to restart the search process in case it gets trapped in a local optimum, consequently giving more opportunities to the algorithm to possibly achieve better solutions within the same computational budget; such kind of approach, briefly discussed in Section

2.3.3, is very common in Evolution Strategies [Auger and Hansen, 2005].

Finally, concerning the artificial scenarios proposed for the empirical assessment of the AOS techniques, it is true that many different and more general settings could have been proposed, such as considering more than two $\mathcal{TV}$ operators, or more complex reward distributions (*e.g.*, taking uniformly drawn values in two intervals centered in the r and R values, or smoother transitions between epochs). But the preliminary motivation was the analysis of the effect of both, the reward level of informativeness R/r and the probability p of receiving a positive reward, on the AOS performance, as will be shown in the empirical comparisons presented in Section 6.3. More complex and realistic reward landscapes shall be considered in further studies, according to the needs of the empirical analysis.

Chapter 6

Experimental Results

Contents

In this Chapter, we present some empirical evaluations of the proposed AOS contributions. The methods are compared between each other, and with other baseline approaches, on diverse benchmark scenarios. Besides the performance evaluation, their sensitivity and robustness in relation to their hyper-parameters is also analyzed.

6.1 Introduction

From the standard MAB *Operator Selection* technique with a *Credit Assignment* based on fitness improvements, up to the latest RMAB that is rewarded based on ranks, several contributions for Adaptive Operator Selection (AOS) have been proposed in Chapter 5. In this Chapter, a comprehensive empirical analysis for each of the AOS methods, resulting from the combination of the proposed *Operator Selection* mechanisms and *Credit Assignment* schemes, will be presented.

Their performances will be assessed and compared in diverse benchmark scenarios, with different characteristics and levels of complexity. Firstly (Section 6.3), the empirical analysis on the artificial scenarios will consider the selection between operators whose rewards come from different pre-defined artificial distributions that are deterministically changed after every ΔT iterations. Then (Section 6.4), experiments on some boolean EA benchmark problems (OneMax, Long K-Path and Royal Road) will be analyzed: in these cases, the AOS schemes are applied to a Genetic Algorithm, automatically selecting between actual mutation and crossover operators, with the rewards coming from the progress attained by the search process in the considered fitness landscapes. Additionally (Section 6.5), the results obtained in the scope of a collaboration with Université d'Angers will be reminded: in this scenario, only the Dynamic Multi-Armed Bandit (DMAB) *Operator Selection* technique (Section 5.3.2) is evaluated, combined with the Compass *Credit Assignment* method (Section 4.3.4). The resulting AOS combination is used by a GA to autonomously select between some crossover, mutation and local search operators, in the context of Boolean Satisfiability (SAT) problems. In the last set of complete empirical comparison,(Section 6.6), the AOS schemes are used within a Differential Evolution algorithm in order to control which of the available mutation strategies should be applied; the results will be assessed in a big set of single-objective continuous benchmark functions.

Prior to all these experiments, a preliminary off-line tuning of the hyper-parameters was done for each AOS technique, in order to promote a fair empirical comparison. The list of hyper-parameters, the range of values tried for each of them, as well as the off-line tuning procedure, will be presented in Section 6.2, together with some general experimental settings. Besides, the specific experimental settings for each scenario will also be detailed in the respective Sections, before the presentation of the results.

In addition to the performance, the sensitivity of each hyper-parameter and the robustness of the AOS techniques with respect to their hyper-parameters will also be analyzed in Section 6.7. Finally, the conclusions and findings gathered from all these empirical data will be discussed in Section 6.8.

6.2 General Experimental Settings

The different AOS combinations proposed have some hyper-parameters that need to be set, as discussed throughout Chapter 5. On each benchmark scenario described in the following, an off-line tuning procedure was preliminarily performed for the hyper-parameters of each technique, for the sake of a fair empirical comparison. A summary of the AOS combinations and their respective hyper-parameters is presented in Section 6.2.1, while Section 6.2.2 describes the values explored for each hyper-parameter, and the off-line tuning procedure used. Finally, Section 6.2.3 overviews the different performance measures and displays that have been used in the diverse empirical comparisons that will be presented in the following.

6.2.1 AOS Combinations and respective Hyper-Parameters

In this Chapter, all the AOS combinations proposed in Chapter 5, namely, the bandit-based MAB, DMAB and SLMAB *Operator Selection* mechanisms, combined with Absolute and Normalized versions of the Instantaneous, Average, and Extreme *Credit Assignment* schemes, as well as the RMAB *Operator Selection* with the rank/comparison-based *Credit Assignment* schemes, are compared with one another on different benchmark scenarios.

Regarding the *Credit Assignment* schemes (Section 5.2), all the schemes except for the Instantaneous one have a common hyper-parameter, the size of the sliding window W, which defines how many operator applications are taken into account to calculate the credit to be assigned to a given operator after its most recent application. It is important to remember that the rank/comparison-based schemes have only one window for all operators, while the other schemes use one window per operator. For the schemes based on the raw values of the fitness improvements, although the fact of normalizing the output or not could be considered as another (boolean) hyper-parameter, schemes employing the Absolute or the Normalized values are separately considered. The rank/comparison-based schemes have an additional parameter, the exponential decay factor of the ranking distribution, referred to as D. Here, again, the choice of use of the fitnesses or the fitness improvements as impact measures for the ranking could be seen as a hyper-parameter, but the schemes implementing each of them are independently analyzed and compared between each other.

On the *Operator Selection* side, the bandit-based schemes, presented in Section 5.3, all have a common hyper-parameter, the scaling factor C. This hyper-parameter defines the balance between the UCB exploration and exploitation terms; in case one of the *Credit Assignment* schemes based on the raw values of fitness improvements is used, C also accounts for the scale of the received rewards, as discussed in Section 5.3.4. Additionally, the DMAB also needs the setting of the threshold γ for the Page-Hinkley change-detection statistical test, used by its restarting mechanism; while the SLMAB requires the definition of its own sliding window size w, used by its update mechanism – however, after some preliminary experiments, this hyper-parameter will be tuned "for free" here, by using the same value than that of the *Credit Assignment* W.

As baseline AOS method for comparison, we consider the probability-based Adaptive Pursuit (AP) *Operator Selection* scheme [Thierens, 2005] (Section 4.4.2), combined with

the same *Credit Assignment* schemes based on the raw values of fitness improvements. AP needs the setting of: the adaptation rate α to control the update of the empirical quality estimates of each operator; the learning rate β, which defines the level of greediness of the winner-take-all strategy for the update of the application rates of each operator; and the minimal application probability of each operator, referred to as p_{min}, in order to avoid inefficient operators to get lost by the process (*i.e.*, have zero probability of being applied), as they might become useful in a further stage of the search. Experiments were also done considering the PM method (Section 4.4.1), but its results will not be neglected here, due to the fact that it was always outperformed (most of the times significantly) by AP.

The other methods used for comparison are: the "Naive" uniform strategy, *i.e.*, the operator to be applied is randomly selected using a uniform distribution, which represents what would be a common choice for a naive user; and the "Oracle" strategy, available only to some of the benchmark problems considered, that represents what would be the optimal behavior with respect to operator selection on the problem at hand. Needless to say, these two latter methods do not have any hyper-parameter to be tuned. Additionally, for the experiments on the boolean benchmark problems, the probabilities of applying each operator were off-line tuned and will be used as a further baseline method, referred to as "Static"; more details will be given within the specific experimental settings for these scenarios (Section 6.4.1).

The lists of the considered *Credit Assignment* and *Operator Selection* schemes, with their corresponding hyper-parameters, are presented in Tables 6.1 and 6.2, respectively; while Table 6.3 summarizes the AOS combinations considered.

Baseline *Credit Assignment* Schemes		**Hyper-Parameters**
Absolute Instantaneous	(AbsIns)	—
Normalized Instantaneous	(NormIns)	
Absolute Average	(AbsAvg)	W Sliding window size
Normalized Average	(NormAvg)	
Proposed *Credit Assignment* Schemes		**Hyper-Parameters**
Absolute Extreme	(AbsExt)	W Sliding window size
Normalized Extreme	(NormExt)	
Decay/Area-Under-Curve	(Decay/AUC)	W Sliding window size
Decay/Fitness-based Area-Under-Curve	(Decay/FAUC)	
Decay/Sum-of-Ranks	(Decay/SR)	D Decay factor
Decay/Fitness-based Sum-of-Ranks	(Decay/FSR)	
NDCG/Area-Under-Curve	(NDCG/AUC)	W Sliding window size
NDCG/Fitness-based Area-Under-Curve	(NDCG/FAUC)	
NDCG/Sum-of-Ranks	(NDCG/SR)	
NDCG/Fitness-based Sum-of-Ranks	(NDCG/FSR)	

Table 6.1: Baseline and proposed *Credit Assignment* methods, and their hyper-parameters

Baseline *Operator Selection* Methods		**Hyper-Parameters**	
Naive Oracle (when available)		—	
Adaptive Pursuit	(AP)	p_{min} α β	Minimal operator probability Adaptation rate Learning or "greediness" rate
Proposed *Operator Selection* Methods		**Hyper-Parameters**	
Multi-Armed Bandit	(MAB)	C	Scaling factor
Dynamic Multi-Armed Bandit	(DMAB)	C γ	Scaling factor Threshold for Page-Hinkley test
Sliding Multi-Armed Bandit	(SLMAB)	C w	Scaling factor Window size (no tune, $w \leftarrow W$)
Rank-based Multi-Armed Bandit	(RMAB)	C	Scaling factor

Table 6.2: Baseline and proposed *Operator Selection* methods, and their hyper-parameters

Credit Assign.				**Op. Selection**	**Credit Assign.**				**Op. Selection**
Abs. Norm.	×	Ins. Avg. Ext.	×	AP MAB DMAB SLMAB	Decay NDCG	×	AUC FAUC SR FSR	×	RMAB
24 fitness-based combinations					8 rank-based combinations				

Table 6.3: List of 32 AOS combinations considered in most experiments

6.2.2 Off-line Tuning of Hyper-Parameters

To promote a fair empirical comparison, it is generally desirable to evaluate the AOS schemes at their best. Accordingly, an off-line tuning was performed preliminarily to every experiment in order to determine, for each AOS combination, the best hyper-parameter configuration. Table 6.4 presents the ranges of values tried for each hyper-parameter, unless stated otherwise.

Briefly, the *Credit Assignment* settings involve 4 configurations for the schemes based on the raw values of fitness improvements and for the rank-based schemes using the NDCG decaying mechanism, while 20 configurations are explored for the other rank-based schemes (using Decay). The *Operator Selection* settings include: 64 possible configurations for AP, 7 for MAB and RMAB, and 49 for DMAB. For the SLMAB, there are 7 possible configurations, except for its combination with the *Instantaneous*, when the *Credit Assignment* window size is set to 1 but all the 4 values are also tried for the sliding window used by its update rule, thus summing up to 28 configurations. The final number of possible configurations for each AOS combination is attained by multiplying the number of configurations for its respective *Credit Assignment* and *Operator Selection* components.

In order to find the optimal values of all hyper-parameters for each AOS combination on each of the analyzed scenarios, rather than a complete factorial Design of Experiments,

Param.	Used by	|values|	Range of Values
p_{min}	AP	4	$\{0, .05, .1, .2\}$
α	AP	4	$\{.1, .3, .6, .9\}$
β	AP	4	$\{.1, .3, .6, .9\}$
C	all MABs	7	$\{0.01, 0.1, 0.5, 1, 5, 10, 100\}$
γ	DMAB	7	$\{10^{-3}, ..., 10^{3}\}$
W	all Credit Assignment	4	$\{10, 50, 100, 500\}$
D	Decay/rank-based Cr. Assign.	5	$\{0.25, 0.5, 0.75, 0.9, 1.0\}$

Table 6.4: Ranges of values tried for the corresponding hyper-parameters

we used the F-Race off-line parameter tuning method [Birattari *et al.*, 2002]. As discussed in Section 3.3.2, the general idea of Racing techniques is to start with all configurations, discarding some of them as soon as there is enough statistical evidence showing that they will not likely be the best one. More specifically, the F-Race applies the Racing paradigm using the *Friedman two-way analysis of variance by ranks* [Conover, 1999] as statistical test to eliminate candidate configurations. In order to enable a fair comparison, as recommended in [Birattari, 2004b], all the experiments in this work use, for each epoch, the same initial population (the "blocking design" concept). Starting from a minimal number of 11 runs, after each run for all configurations, the elimination of inefficient configurations is performed with the statistical test being applied at a confidence level of 95%. Although 11 runs might be excessive (*e.g.*, [Birattari, 2004a] recommends one run over each instance), we prefer to be conservative here, in order to be sure that the methods are really compared at their best, specially because there is a considerable variance in the results of some of the benchmark problems. The procedure is stopped when a single configuration remains, or when all "survivors" have been run on the maximal number of runs, set to 50 in these experiments. In the latter case, the retained configuration is the one with the best mean amongst the survivors, as done in [Birattari *et al.*, 2002] (although different alternatives could be used, *e.g.*, in a critical situation the configuration with best worst case could be considered). In all cases, 50 runs are launched for the retained configuration and the results presented in this paper are based on statistics over these runs, unless stated otherwise.

It is worth noting that other off-line tuning methods than the F-Race could have been tried, notably the methods that iteratively refine the set of candidate configurations, such as the Iterated F-Race [Balaprakash *et al.*, 2007], the Sequential Parameter Optimization [Bartz-Beielstein *et al.*, 2005], and others, surveyed in Section 3.3.2. However, the objective here was rather to simply do some tuning of the hyper-parameters while being more computationally efficient. Although we acknowledge that the use of different off-line tuning techniques, or different sets of candidate configurations tuned by the same F-Race, could eventually lead to different winners on some benchmark scenarios, we believe that this issue does not affect the global conclusions gathered from the diverse benchmark situations that will be considered in the following.

6.2.3 Performance Indicators and Results Presentation

Several complementary views and indicators for analyzing the AOS performance will be used on each benchmark scenario. The main ones will be summarized now, distinguishing between off-line and on-line performance presentations.

Off-line Performance

The diverse benchmark scenarios considered in the following use different indicators to evaluate the performance of the AOS schemes. These indicators will be described in the corresponding Sections presenting the specific Experimental Settings for each scenario. The off-line performance measures are compared in a Table, for each analyzed scenario, in which each cell corresponds to a *Credit Assignment* and an *Operator Selection*, and indicates:

- the average and standard deviation of the performance measure at hand (according to the type of scenario); and
- the values of the best hyper-parameter configuration determined after the F-Race procedure, as described in Section 6.2.2.

Besides the numerical results, two different kinds of statistical comparison are shown in the same Table (see Table 6.5 for instance). The first comparison is the global one: the best performance achieved between all the considered AOS techniques is presented in bold-face with grey background, like **this**; the results which are statistically equivalent, according to at least one of both unsigned Wilcoxon rank sum and Kolmogorov-Smirnov non-parametric tests applied at confidence level 90%, are displayed with a grey background. Small result variations among the different AOS schemes thus translate like many grey cells. The second comparison takes place in the scope of each *Operator Selection* technique: the best performance achieved by it using one of the available *Credit Assignment* schemes is marked with a ★ symbol; accordingly, all the schemes within the same *Operator Selection* technique that obtained equivalent performance, according to the same statistical tests, are marked with a ▲ symbol. Finally, the caption of the table finally indicates the performances of the Naive uniform strategy, and of the Oracle and Static strategies (whenever available).

In some scenarios, however, given the high dispersion of the results, it is not meaningful to present only the averages and standard deviations, *i.e.*, no significant difference can be found, in terms of performance, between most of the AOS schemes. Because of this, in order to be able to figure out the difference of performance between the techniques, the empirical distribution of the results over 50 runs is presented, depicted using Empirical Cumulative Distribution Functions (ECDFs): for each level of performance (on x-axis) the percentage of runs reaching this score is indicated (on y-axis); see, *e.g.*, Fig. 6.15b. The x-axis is limited by the average performance of the Naive uniform approach: thus, the y value attained by the corresponding ECDF curve at the right border of the plot represents the percentage of runs of the method under assessment that performed better than the Naive uniform baseline. ECDFs are preferred in lieu of standard box-plot

diagrams, mainly because they display of the full distribution, consequently enabling a fine-grained comparison of the different schemes, accounting for the fact that one scheme might outperform another scheme with regard to some quantile performance, although being outperformed with respect to the average or median value.

ECDFs are also used to assess the sensitivity of the schemes with respect to their hyper-parameters. More precisely, ECDFs aggregating series of runs corresponding to several hyper-parameter configurations will be presented for each AOS. These plots graphically show how fast the performance degrades when departing from the optimal parameter configuration.

On-line Performance

The off-line performance, however, does not tell whether an AOS fails to detect the best operator due to an excess of exploration, or exploitation. In the former case, the AOS fails to stick to the best operator after the change; in the latter case, it fails to swiftly adapt whenever a change occurs.

For this reason, the on-line performance of the AOS schemes with respect to operator selection is also presented for some benchmark scenarios. The on-line performance plots (or behavior plots, *e.g.*, Fig. 6.2) depict, for each operator, its instant selection rate along time. For the sake of a smoother presentation, given the high variation of behavior between the many runs, the selection rates plotted represent the average of every 50 time steps for each run, further averaged over 50 runs. Additionally, on all such plots for DMAB, the small peaks below the x-axis indicate the frequency of restarts after the triggering of the PH change-detection statistical test (also averaged every 50 steps, over 50 runs).

6.3 On Artificial Scenarios

In order to assess the proposed and the baseline AOS combinations in controlled environments, *i.e.*, in benchmark problems in which the expected behavior is exactly known, experiments were done on the artificial scenarios described in Section 5.4. The specific experimental settings for these experiments, in complement to the general settings presented in Section 6.2, will be described in Section 6.3.1. The empirical comparison of the AOS schemes on the Uniform, Boolean and Outlier scenarios will be surveyed in Section 6.3.2, while Section 6.3.3 will present the results involving two different $\mathcal{ART}$ instances. Finally, Section 6.3.4 will present a discussion about the highlights of these experiments. The results that will be analyzed here were partially published in [Da Costa *et al.*, 2008; Fialho *et al.*, 2010a; Fialho *et al.*, 2010c].

6.3.1 Experimental Settings

On these artificial scenarios, the most natural performance indicator is the total gain brought by an AOS scheme, referred to as the Total Cumulated Reward (TCR), computed as the sum of the rewards gathered by the algorithm over the complete run.

Besides the average and standard deviation of the TCR, a related but not equivalent indicator, the percentage of times the best operator is selected, is also presented for an illustrative purpose in the comparison tables (see, *e.g.*, Table 6.7); it will be referred to as $p(best)$ in the remainder of this text. It is important to note that two techniques might present similar $p(best)$, although presenting very different TCR performance, due to the difference between the sub-optimal choices done by each of them, the so-called error costs, which are not taken into account by this measure.

In all cases, the reward distributions are modified every ΔT time steps. The epoch lengths considered are $\Delta T \in \{50, 200, 500, 2000\}$, *i.e.*, ranging from very fast to very slow dynamics. Ten epochs are considered, unless stated otherwise. Although being already defined in the corresponding Sections 5.4.2 and 5.4.3, the sequences used for the exchanges of reward distributions are reminded here: for the Uniform, Boolean and Outlier scenarios, the sequence is $41203 \mapsto 01234 \mapsto 24301 \mapsto 12043 \mapsto 41230 \mapsto 31420 \mapsto 04213 \mapsto 23104 \mapsto 14302 \mapsto 40213$; and for the $\mathcal{ART}$ scenarios, $01 \mapsto 01 \mapsto 10 \mapsto 01 \mapsto 10 \mapsto 10 \mapsto 10 \mapsto 01 \mapsto 01 \mapsto 10$.

An alternative view of off-line performance is also shown for each artificial scenario: how the performance (in terms of TCR and $p(best)$) of each *Operator Selection* technique, with its best *Credit Assignment* scheme found by the off-line tuning, scales with respect to the different epoch lengths considered (see, for instance, Figure 6.1). The motivation is not to compare the results on the different scenarios, but rather to compare the "trend" of the scaling of the performance of the AOS combinations on the different epoch lengths.

6.3.2 Results on Uniform, Boolean and Outlier Scenarios

The Uniform, Boolean and Outlier scenarios involve 5 operators, with rewards coming from different distributions, as described in Sections 5.4.1 and 5.4.2. Empirical results on each of these scenarios will be separately analyzed in the following.

Uniform scenario

On the Uniform scenario, results are very clear and almost in total accordance with respect to all the four different epoch lengths considered. Tables 6.5 and 6.6 present, respectively, the results obtained on $\Delta T \in \{50, 200\}$ and $\Delta T \in \{500, 2000\}$.

Given the high informativeness of the rewards received with respect to the quality of the operators, and their steady-stateness during each epoch, it becomes reasonably easy to detect which is the current best operator on this scenario. In the worst case, as the reward distributions of subsequent operators partially overlap between each other, the second best operator might occasionally be considered to be the best one; but the error cost is always very small for the same reason, therefore it does not greatly affect the Total Cumulated Reward (TCR). The difference in performance thus lies mainly in how fast the AOS techniques are able to adapt to new situations, whenever a change occurs.

Accordingly, using Average or Extreme *Credit Assignment* schemes will always delay, in W operator applications at most, the perception that the situation has changed; this is a tentative explanation for the fact that both Absolute and Normalized versions of the

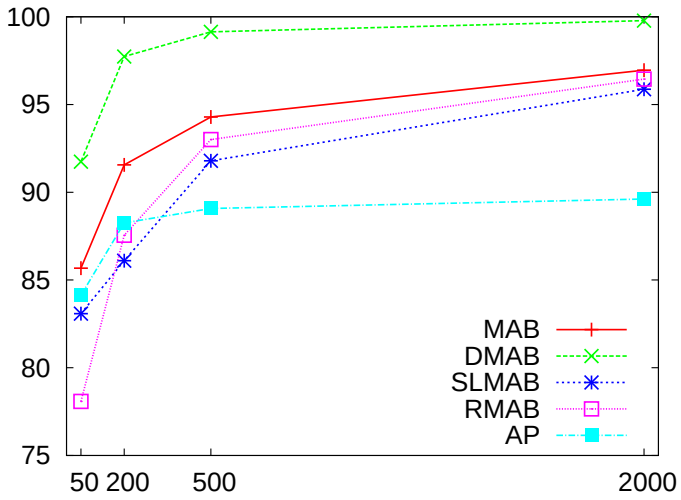
Instantaneous *Credit Assignment* scheme are clearly the best options for all the *Operator Selection* methods tried with them. This interpretation is supported by the output of the Racing process for each of the AOS combinations using the Average and the Extreme schemes: the best configuration retained for all of them has the same sliding window size $W = 10$, *i.e.*, the lowest value in the range tried for this hyper-parameter (Table 6.4).

Between the rank-based *Credit Assignment* schemes, the AUC is the best option. The fact that a linear decay (*i.e.*, $D = 1$) is retained as best configuration might also be related to the subtle overlap between the rewards received by the different operators: in such a case, there is no need of a strong decaying factor to differ between them.

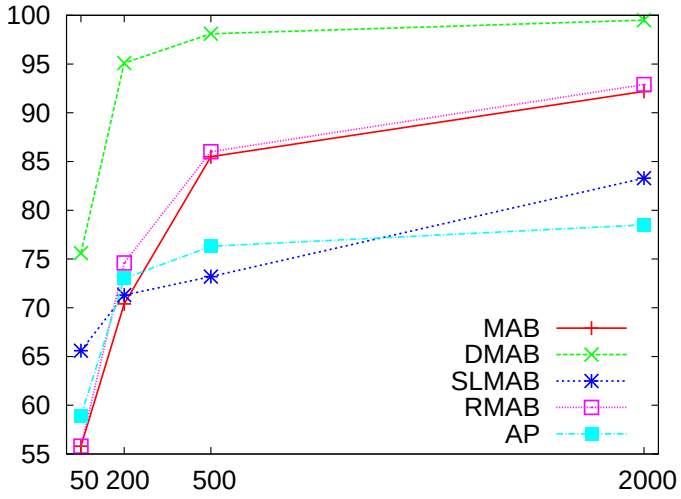
On the *Operator Selection* side, the clear winner is DMAB, which, combined with the Absolute Instantaneous (AbsIns) *Credit Assignment* scheme, significantly outperforms all the other AOS combinations on all epochs, except for the same DMAB with Normalized Instantaneous (NormIns) *Credit Assignment* in three out of four cases. Indeed, for the longest epoch ($\Delta T = 2000$), the AbsIns-DMAB AOS technique obtains a TCR equivalent to 99.7% of the Oracle TCR, selecting the best operator in around 99.5% of the applications. This outstanding performance is explained by the well-calibrated PH test (as shown in Figures 6.2a and 6.2b); although using a very sensitive hyper-parameter (see sensitivity analysis in Section 6.7), the restarting mechanism is very efficient on this kind of situation in which the qualities of the operators change abruptly. For faster dynamics (smaller ΔT), the performance is gracefully degraded: shorter the steady-state epoch, less negligible becomes the price to be paid by the extra exploration needed after each restart. But, still, DMAB remains the clear winner with respect to TCR and $p(best)$ for all epoch lengths, as shown in Figure 6.1.

The standard MAB is the second best *Operator Selection* technique on this scenario, achieving a performance slightly (but significantly, given the small standard deviations) inferior than that of DMAB, for all epoch lengths. Its combination with either AbsIns or NormIns *Credit Assignment* schemes, although being the simplest bandit-based AOS combination considered, is able to achieve up to 97% of the Oracle TCR for $\Delta T = 2000$, selecting the best operator in around 92% of the times (Figure 6.2c); its well-tuned EvE balance enables the discovery of the new situation by paying the minimal price with respect to exploration. The SLMAB is also able to swiftly follow the dynamics of the scenario, employing the same level of exploration; but it achieves similar performance than that of MAB only because of the low error cost, as it is not able to differ well between the best and the second best operators in some of the epochs (Figure 6.2d); this explains why its $p(best)$ is much lower than that of MAB, although obtaining a similar TCR. The AUC-RMAB combination with linear decay achieves very similar performance (Figure 6.2e) than that of MAB; given the very low value retained for its scaling factor C, it becomes clear that all the adaptation is in fact done on the *Credit Assignment* side: an impact measure brought by the application of one operator might affect the whole ranking distribution, consequently affecting the quality estimate of all operators, as discussed in Section 5.2.4.

Conversely, the baseline probability-based Adaptive Pursuit (AP) method does not follow the same trend, being significantly outperformed by all bandit-based schemes on the two longer epochs (see Figure 6.1). This limitation is blamed on an excess of exploration (lower bounded by the p_{min}=0.05 parameter), as shown in Figure 6.2f. It is worth noticing



(a) % max TCR w.r.t. epoch length



(b) p(best) w.r.t. epoch length

Figure 6.1: Scaling of mean performance (TCR above, and $p(best)$ below) in relation to the epoch length ΔT, for each *Operator Selection* technique with its best *Credit Assignment* scheme, on the Uniform scenario.

that p_{min}=0 was also tried in the parameter tuning process (Table 6.4), but achieved lower performance than that of p_{min}=0.05; the value used in fact translates into up to 25% of exploration trials in this case, as p_{min} refers to the minimal level of exploration for *each* operator. Finally, all AOS combinations succeed in performing significantly better than the Naive uniform baseline method.

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	2077 ± 55 ★ C10W50 Pb: 65.6 ± 4.9	2171 ± 36 C1G1W1 Pb: 66.9 ± 4.8	2120 ± 23 C5W1 Pb: 63.5 ± 1.7	2103 ± 35 ★ P.05A.9B.9W1 Pb: 58.9 ± 5.6
NormIns	1957 ± 109 C1W10 Pb: 44.8 ± 11.6	**2294 ± 30 ★** **C.01G.1W1** **Pb: 75.6 ± 5.3**	2142 ± 61 ★ C.5W1 Pb: 55.8 ± 8.6	2077 ± 43 P.05A.9B.9W1 Pb: 56.6 ± 5.5
AbsAvg	1926 ± 103 C5W10 Pb: 45.2 ± 10.7	2084 ± 49 C.5G1W10 Pb: 48.9 ± 6.0	1915 ± 40 C1W10 Pb: 41.7 ± 4.6	2030 ± 70 P0A.9B.9W10 Pb: 39.8 ± 6.4
NormAvg	1896 ± 102 C1W10 Pb: 41.6 ± 11.3	1907 ± 52 C.5G10W10 Pb: 34.1 ± 5.3	1907 ± 52 C.5W10 Pb: 34.1 ± 5.3	1906 ± 100 P0A.1B.9W10 Pb: 33.3 ± 8.0
AbsExt	1879 ± 97 C10W100 Pb: 38.8 ± 6.6	2120 ± 33 C.5G1W10 Pb: 70.0 ± 3.4	1884 ± 46 C1W10 Pb: 40.8 ± 5.8	1892 ± 43 P.05A.9B.9W10 Pb: 46.7 ± 6.3
NormExt	1800 ± 105 C1W10 Pb: 38.4 ± 9.3	1990 ± 54 C.01G.1W10 Pb: 50.0 ± 7.2	1949 ± 39 C.5W10 Pb: 51.4 ± 3.3	1868 ± 41 P.05A.9B.9W10 Pb: 43.2 ± 5.9
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	1952 ± 19 ★ C.5D1W10 Pb: 55.8 ± 1.3	1947 ± 18 ▲ C.5D.75W10 Pb: 56.4 ± 1.5	1927 ± 16 C.5W10 Pb: 55.0 ± 1.8	1937 ± 17 C.5W10 Pb: 55.7 ± 1.6

(a) Results on the Uniform scenario, for ΔT=50 (Naive TCR: 1500, Optimal TCR: 2500)

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	8530 ± 329 ▲ C5W10 Pb: 56.7 ± 11.8	**9773 ± 117 ★** **C.1G10W1** **Pb: 95.1 ± 1.5**	9145 ± 37 ▲ C5W1 Pb: 78.6 ± 0.7	8826 ± 73 ★ P.05A.6B.9W1 Pb: 73.0 ± 2.5
NormIns	8610 ± 106 ★ C5W500 Pb: 71.3 ± 1.9	9699 ± 136 ▲ C.01G.001W1 Pb: 89.7 ± 6.8	9156 ± 154 ★ C.5W1 Pb: 70.4 ± 6.3	8806 ± 68 ▲ P.05A.9B.9W1 Pb: 72.6 ± 2.2
AbsAvg	8355 ± 346 C5W10 Pb: 54.3 ± 12.2	9066 ± 108 C1G1W10 Pb: 67.6 ± 5.1	8769 ± 64 C5W10 Pb: 71.2 ± 1.7	8561 ± 300 P0A.9B.9W10 Pb: 43.9 ± 9.3
NormAvg	7909 ± 427 C1W10 Pb: 36.6 ± 11.0	8643 ± 138 C.5G10W10 Pb: 49.5 ± 4.7	8643 ± 138 C.5W10 Pb: 49.5 ± 4.7	8228 ± 121 P.05A.9B.9W10 Pb: 50.7 ± 4.5
AbsExt	8128 ± 323 C5W10 Pb: 43.2 ± 10.6	9526 ± 39 C.5G10W10 Pb: 90.2 ± 0.8	8867 ± 43 C5W10 Pb: 74.6 ± 0.8	8626 ± 85 P.05A.9B.9W10 Pb: 70.7 ± 2.7
NormExt	7668 ± 393 C1W10 Pb: 34.3 ± 11.2	9236 ± 159 C.1G.001W10 Pb: 77.8 ± 6.4	8908 ± 142 C.5W10 Pb: 69.8 ± 5.6	8593 ± 94 P.05A.9B.9W10 Pb: 69.6 ± 3.0
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	8755 ± 87 ★ C.1D1W50 Pb: 74.6 ± 2.5	8469 ± 92 C.1D.5W50 Pb: 68.9 ± 3.7	8469 ± 92 C.1W50 Pb: 68.9 ± 3.7	8469 ± 92 C.1W50 Pb: 68.9 ± 3.7

(b) Results on the Uniform scenario, for ΔT=200 (Naive TCR: 6000, Optimal TCR: 10000)

Table 6.5: Results on the Uniform scenario for $\Delta T \in \{50, 200\}$

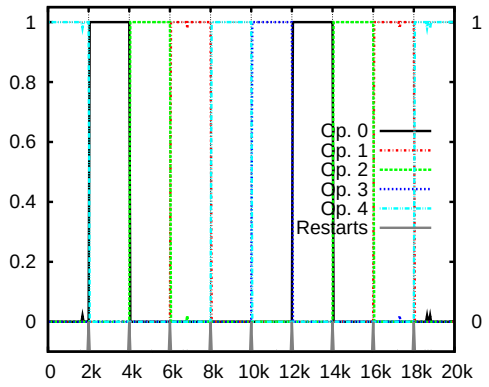
Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	22948 ± 678 ★ C5W10 Pb: 73.2 ± 11.8	**24785 ± 41 ★** **C.1G10W1** **Pb: 98.1 ± 0.2**	23572 ± 66 ★ C5W1 Pb: 85.5 ± 0.5	22270 ± 91 ★ P.05A.6B.9W1 Pb: 76.3 ± 1.2
NormIns	21041 ± 637 C5W500 Pb: 63.6 ± 6.6	24448 ± 400 C.01G.001W1 Pb: 90.9 ± 8.0	23294 ± 377 C.5W1 Pb: 73.3 ± 7.1	22212 ± 97 ▲ P.05A.9B.9W1 Pb: 75.4 ± 1.1
AbsAvg	22659 ± 631 C5W10 Pb: 69.8 ± 11.0	23494 ± 222 C1G1W10 Pb: 78.5 ± 4.0	23199 ± 114 C5W10 Pb: 82.8 ± 1.0	21778 ± 119 P.05A.9B.9W10 Pb: 68.2 ± 1.9
NormAvg	20712 ± 1057 C1W10 Pb: 42.6 ± 12.8	22777 ± 253 C.5G.01W10 Pb: 71.4 ± 3.9	22502 ± 221 C1W10 Pb: 68.2 ± 3.4	21671 ± 133 P.05A.9B.9W10 Pb: 67.2 ± 2.0
AbsExt	21578 ± 776 C5W10 Pb: 50.5 ± 14.1	24509 ± 48 C.5G10W10 Pb: 95.9 ± 0.4	23275 ± 69 C5W10 Pb: 83.7 ± 0.5	22085 ± 105 P.05A.9B.9W10 Pb: 75.7 ± 1.3
NormExt	20471 ± 1526 C1W10 Pb: 43.0 ± 14.4	23882 ± 355 C.1G.001W10 Pb: 84.4 ± 6.1	23109 ± 366 C.5W10 Pb: 75.7 ± 5.7	22063 ± 114 P.05A.9B.9W10 Pb: 75.5 ± 1.4
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	23251 ± 117 ★ C.1D1W50 Pb: 86.0 ± 1.2	22804 ± 134 C.1D.5W50 Pb: 81.8 ± 2.6	22804 ± 134 C.1W50 Pb: 81.8 ± 2.6	22804 ± 134 C.1W50 Pb: 81.8 ± 2.6

(a) Results on the Uniform scenario, for ΔT=500 (Naive TCR: 15000, Optimal TCR: 25000)

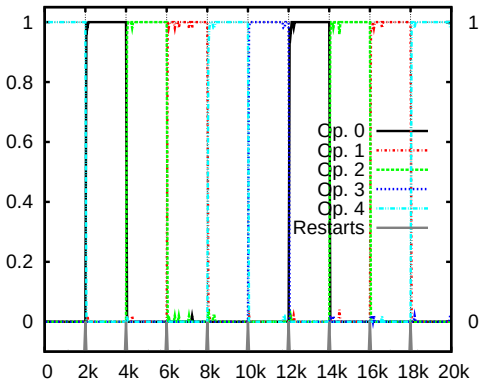
Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	95872 ± 2745 ★ C5W10 Pb: 83.3 ± 13.1	**99781 ± 72 ★** **C.1G10W1** **Pb: 99.5 ± 0.0**	96954 ± 128 ★ C5W1 Pb: 92.2 ± 0.3	89617 ± 156 ★ P.05A.3B.9W1 Pb: 78.5 ± 0.5
NormIns	83221 ± 6417 C1W10 Pb: 45.3 ± 17.2	98731 ± 1583 ▲ C.01G.001W1 Pb: 94.2 ± 7.9	95329 ± 916 C1W1 Pb: 84.5 ± 3.3	89534 ± 159 ▲ P.05A.3B.9W1 Pb: 78.3 ± 0.5
AbsAvg	95135 ± 663 C10W10 Pb: 86.2 ± 2.4	97625 ± 1105 C1G10W10 Pb: 89.8 ± 5.4	96332 ± 330 C5W10 Pb: 91.1 ± 0.7	89242 ± 174 P.05A.9B.9W10 Pb: 76.9 ± 0.6
NormAvg	87364 ± 4064 C1W10 Pb: 51.0 ± 13.1	95958 ± 718 C.5G.001W10 Pb: 84.5 ± 3.5	93741 ± 912 C1W10 Pb: 77.4 ± 3.7	89137 ± 192 P.05A.9B.9W10 Pb: 76.7 ± 0.7
AbsExt	94716 ± 613 C10W10 Pb: 85.6 ± 1.9	99492 ± 79 C.5G10W10 Pb: 98.9 ± 0.1	96658 ± 99 C5W10 Pb: 91.7 ± 0.2	89560 ± 154 ▲ P.05A.9B.9W10 Pb: 78.9 ± 0.4
NormExt	85092 ± 6055 C1W10 Pb: 48.8 ± 15.1	97103 ± 1444 C.1G.001W10 Pb: 87.2 ± 7.3	95044 ± 656 C1W10 Pb: 85.5 ± 2.1	89540 ± 161 ▲ P.05A.9B.9W10 Pb: 78.8 ± 0.4
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	96449 ± 231 ★ C.1D1W100 Pb: 92.9 ± 0.6	95501 ± 234 C.1D.5W100 Pb: 90.9 ± 1.1	95501 ± 234 C.1W100 Pb: 90.9 ± 1.1	95501 ± 234 C.1W100 Pb: 90.9 ± 1.1

(b) Results on the Uniform scenario, for ΔT=2000 (Naive TCR: 60000, Optimal TCR: 100000)

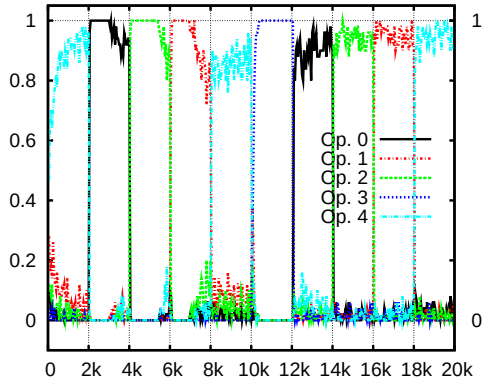
Table 6.6: Results on the Uniform scenario for $\Delta T \in \{500, 2000\}$



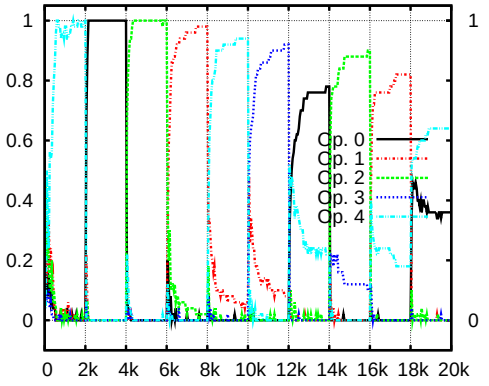
(a) AbsIns-DMAB (C.1 γ10 W1)



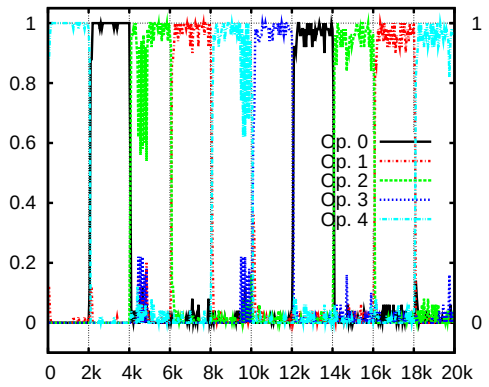
(b) AbsExt-DMAB (C.5 γ10 W10)



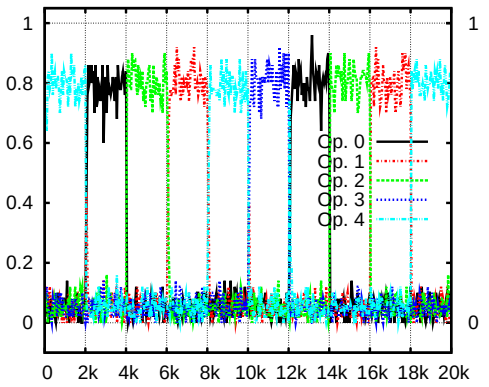
(c) AbsIns-MAB (C5 W1)



(d) AbsIns-SLMAB (C5 W10)



(e) Decay/AUC-RMAB (C.1 D1 W100)



(f) AbsIns-AP (P.05 A.3 B.9 W1)

Figure 6.2: Behavior of DMAB, MAB, SLMAB, RMAB and AP, combined with their best *Credit Assignment* schemes, on the Uniform scenario with $\Delta T = 2000$. The outstanding performance of DMAB, using either AbsInst or AbsExt *Credit Assignment* schemes, is achieved by the fact that restarts are perfectly triggered by the PH test in the transitions, as indicated by the small peaks below the x-axis of the corresponding plots.

Boolean scenario

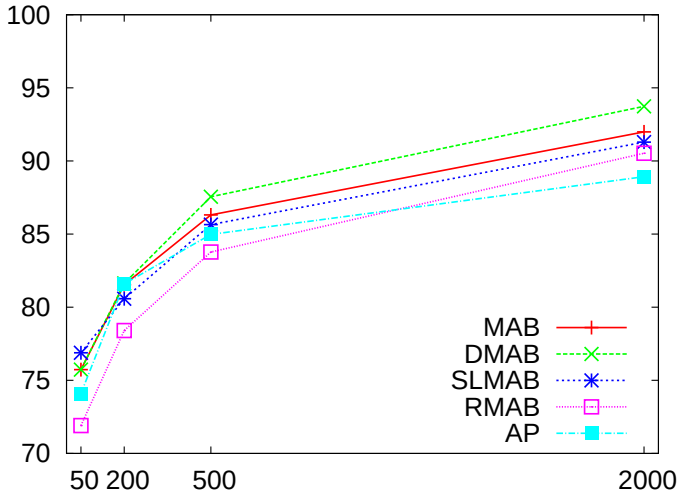
Differently from the Uniform scenario, in the Boolean scenario it is not the values of the rewards that inform which is the best operator, but rather the frequency with which each operator is rewarded: this makes this scenario a very difficult one for AOS, as discussed in Section 5.4.2. Tables 6.7 and 6.8 present, respectively, the results obtained on $\Delta T \in \{50, 200\}$ and $\Delta T \in \{500, 2000\}$, confirming this assumption.

For instance, with $\Delta T = 50$, only 47% of good choices are made by SLMAB, the best method in this case, and less than 40% by all others. When ΔT increases, the overall performance of all *Operator Selection* techniques (with their corresponding best *Credit Assignment* scheme) in relation to both TCR and $p(best)$ measures, gradually increases accordingly, as shown in Figure 6.3. Anyway, the maximum $p(best)$ attained in the longest epoch is 80%, while in the Uniform scenario rates up to 99.5% were found.

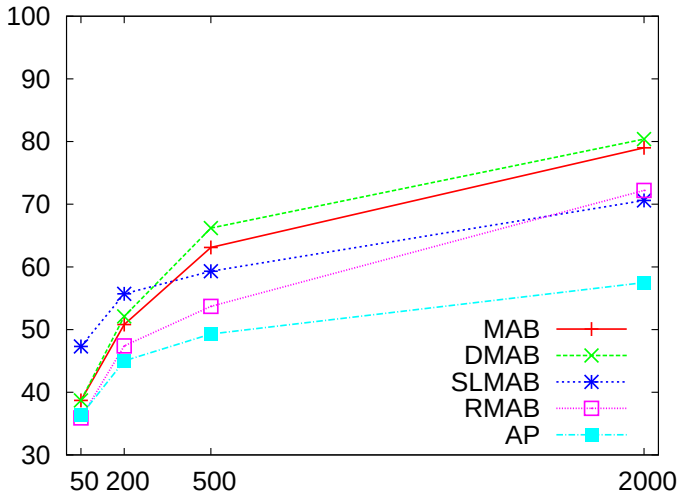
Concerning the *Operator Selection* techniques, for the shortest epoch, SLMAB achieves the best TCR, but its performance is statistically equivalent to all the others. Starting from $\Delta T = 200$, the DMAB takes the lead, with the gap between its performance and those of the others increasing, up to the longest epoch, in which the winner configuration for DMAB is significantly better than all the other techniques. It is important to note that, as the values of the rewards received are always the same (= 10), very few restarts are done by DMAB (see, *e.g.*, Figures 6.4a and 6.4b); hence, the performance of the standard MAB is very similar to the DMAB performance, being significantly different only for the longest epoch. The other three *Operator Selection* techniques, namely, SLMAB, RMAB and the baseline AP, present equivalent but inferior performance for all epoch lengths, except for the longest one, in which SLMAB is significantly better than AP, but still equivalent to RMAB.

By analyzing these behavior plots for $\Delta T = 2000$ in Figure 6.4, it can be seen that, except for the overall winner NormIns-DMAB (Figure 6.4a) and the second best NormIns-MAB (Figure 6.4c), the performance of the other bandit-based AOS combinations is hindered by a lack of further exploitation of the best operator: in around 20% of the trials, a sub-optimal operator is applied (see Figures 6.4d and 6.4e, respectively, for the behavior of SLMAB and RMAB). For AP, although the p_{min} value is set to 0 in this case, its low performance is explained by a failure in identifying the best 'operator' in all cases where the best operator becomes the second best (Figure 6.4f), possibly due to the high inertia of the two-tiered update of empirical quality estimates employed by this method, as discussed in Section 4.4.2.

For the *Credit Assignment*, as for the Uniform scenario, the best scheme is the Instantaneous one, both Absolute and Normalized alternatives performing equivalently in almost all cases. The Average-based schemes performs almost as good; while the Extreme-based ones are outperformed by far: as the only values that are taken here are 0 or 10, it becomes difficult for the Extreme reward to distinguish among operators. Along the same lines, a tentative of interpretation for the inefficiency of the rank-based *Credit Assignment* schemes (no matter the decaying factor used), is that the only two possible values for the rewards do not enable enough granularity in the ranking distribution, consequently resulting in similar qualities to the operators.



(a) % max TCR w.r.t. epoch length



(b) p(best) w.r.t. epoch length

Figure 6.3: Scaling of mean performance (TCR above, and $p(best)$ below) in relation to the epoch length ΔT, for each *Operator Selection* technique with its best *Credit Assignment* scheme, on the Boolean scenario.

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	**1922 ± 162 ★** **C10W50** **Pb:47.3 ± 10.0**	1893 ± 136 ★ C5G1000W1 Pb: 38.7 ± 7.7	1893 ± 136 ★ C5W1 Pb: 38.7 ± 7.7	1798 ± 148 ▲ P0A.3B.9W1 Pb: 29.8 ± 9.2
NormIns	1922 ± 162 ▲ C1W50 Pb: 47.3 ± 10.0	1890 ± 151 ▲ C.5G10W1 Pb: 38.9 ± 7.2	1890 ± 151 ▲ C.5W1 Pb: 38.9 ± 7.2	1852 ± 144 ★ P0A.1B.9W1 Pb: 36.3 ± 9.2
AbsAvg	1855 ± 182 ▲ C5W10 Pb: 37.4 ± 9.7	1793 ± 132 C.5G1W10 Pb: 31.3 ± 7.0	1735 ± 132 C5W10 Pb: 29.6 ± 7.3	1727 ± 116 P.05A.6B.9W10 Pb: 28.5 ± 6.7
NormAvg	1743 ± 169 C1W10 Pb: 31.4 ± 11.8	1775 ± 153 C.01G.1W10 Pb: 27.7 ± 9.3	1764 ± 127 C1W10 Pb: 29.1 ± 6.5	1723 ± 130 P.05A.9B.9W10 Pb: 27.4 ± 8.2
AbsExt	1639 ± 168 C10W10 Pb: 23.6 ± 10.9	1679 ± 162 C5G100W10 Pb: 25.6 ± 6.2	1674 ± 173 C5W10 Pb: 25.4 ± 5.9	1656 ± 158 P0A.6B.6W10 Pb: 22.0 ± 7.1
NormExt	1639 ± 168 C1W10 Pb: 23.6 ± 10.9	1670 ± 173 C.5G100W10 Pb: 25.6 ± 5.9	1670 ± 173 C.5W10 Pb: 25.6 ± 5.9	1693 ± 128 P0A.3B.9W10 Pb: 23.4 ± 8.4
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	1798 ± 135 ★ C.5D1W50 Pb: 35.9 ± 6.6	1716 ± 119 C1D.5W50 Pb: 30.8 ± 4.6	1775 ± 127 ▲ C1W500 Pb: 33.7 ± 4.7	1724 ± 116 ▲ C1W50 Pb: 31.2 ± 4.4

(a) Results on the Boolean scenario, for ΔT=50 (Naive TCR: 1500, Optimal TCR: 2500)

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	8058 ± 427 ★ C10W100 Pb: 55.7 ± 10.6	8154 ± 348 ▲ C5G1000W1 Pb: 50.8 ± 8.6	8154 ± 348 ★ C5W1 Pb: 50.8 ± 8.6	7966 ± 249 P.05A.1B.6W1 Pb: 47.5 ± 6.1
NormIns	8058 ± 427 ▲ C1W100 Pb: 55.7 ± 10.6	**8162 ± 356 ★** **C.5G10W1** **Pb: 52.1 ± 8.9**	8152 ± 364 ▲ C.5W1 Pb: 51.5 ± 8.9	8160 ± 393 ★ P0A.1B.9W1 Pb: 45.0 ± 10.4
AbsAvg	7979 ± 380 ▲ C5W10 Pb: 46.5 ± 9.6	8063 ± 360 ▲ C1G10W10 Pb: 45.2 ± 8.6	7921 ± 313 C5W10 Pb: 45.7 ± 6.2	7871 ± 271 P.05A.9B.6W10 Pb: 45.0 ± 5.0
NormAvg	7536 ± 494 C1W10 Pb: 32.5 ± 11.0	8033 ± 416 ▲ C.01G.001W10 Pb: 43.3 ± 11.0	7769 ± 376 C1W10 Pb: 39.2 ± 8.9	7817 ± 249 P.05A.9B.9W10 Pb: 44.4 ± 5.8
AbsExt	7414 ± 420 C10W10 Pb: 30.5 ± 9.4	7475 ± 520 C1G10W10 Pb: 33.0 ± 10.7	7402 ± 429 C5W10 Pb: 31.3 ± 5.5	7477 ± 526 P0A.9B.9W10 Pb: 31.4 ± 9.7
NormExt	7414 ± 420 C1W10 Pb: 30.5 ± 9.4	7526 ± 480 C.01G1W10 Pb: 31.7 ± 11.4	7409 ± 437 C.5W10 Pb: 31.6 ± 5.7	7611 ± 358 P0A.1B.9W10 Pb: 31.3 ± 8.6
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	7841 ± 289 ★ C.5D1W100 Pb: 47.4 ± 7.8	7476 ± 361 C.5D.75W50 Pb: 36.1 ± 8.9	7580 ± 246 C.5W100 Pb: 40.6 ± 3.2	7466 ± 267 C1W100 Pb: 42.1 ± 5.0

(b) Results on the Boolean scenario, for ΔT=200 (Naive TCR: 6000, Optimal TCR: 10000)

Table 6.7: Results on the Boolean scenario for $\Delta T \in \{50, 200\}$

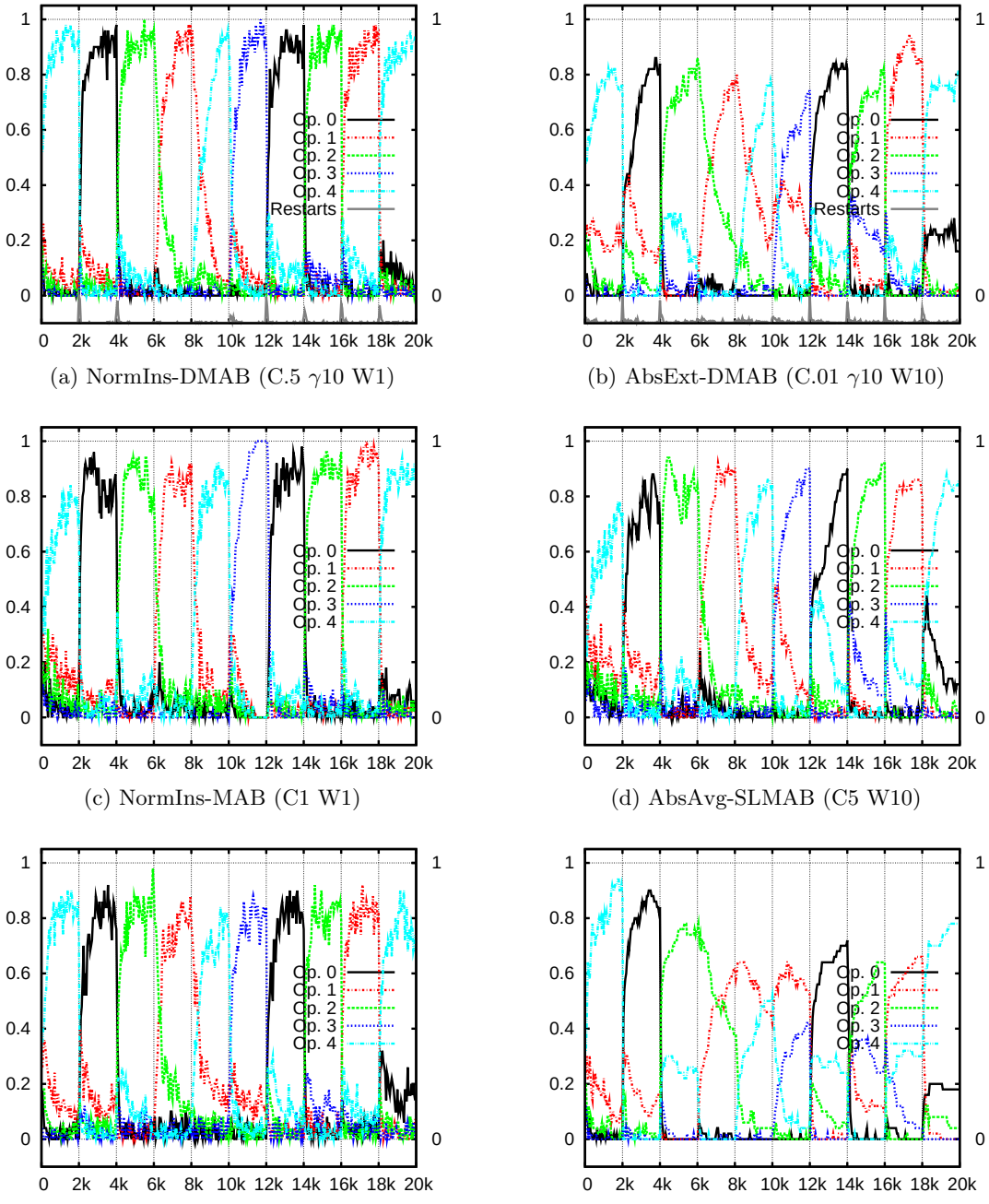
Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	21234 ± 751 ▲ C5W10 Pb: 56.5 ± 9.5	**21888 ± 611 ★** **C5G100W1** **Pb: 66.2 ± 5.4**	21532 ± 834 ▲ C5W1 Pb: 63.3 ± 6.4	20783 ± 1098 ▲ P0A.3B.9W1 Pb: 46.5 ± 10.9
NormIns	21234 ± 751 ▲ C.5W10 Pb: 56.5 ± 9.5	21740 ± 658 ▲ C.5G10W1 Pb: 63.6 ± 6.5	21578 ± 817 ★ C.5W1 Pb: 63.1 ± 7.1	21245 ± 983 ★ P0A.1B.9W1 Pb: 49.3 ± 12.6
AbsAvg	21412 ± 531 ★ C5W10 Pb: 59.3 ± 6.7	21588 ± 604 ▲ C5G100W10 Pb: 62.5 ± 5.8	21106 ± 579 C10W10 Pb: 61.2 ± 4.6	20406 ± 519 P.05A.1B.9W10 Pb: 51.3 ± 5.6
NormAvg	19883 ± 1003 C1W10 Pb: 36.7 ± 11.6	21495 ± 591 ▲ C.01G.1W10 Pb: 54.6 ± 8.7	20711 ± 752 C1W10 Pb: 48.4 ± 7.9	20263 ± 445 P.05A.1B.9W10 Pb: 50.2 ± 4.7
AbsExt	19852 ± 926 C10W10 Pb: 38.2 ± 10.8	20515 ± 854 C.1G10W10 Pb: 41.4 ± 9.5	19649 ± 751 C5W10 Pb: 36.5 ± 6.5	19728 ± 1144 P0A.9B.6W10 Pb: 35.6 ± 12.9
NormExt	19852 ± 926 C1W10 Pb: 38.2 ± 10.8	20492 ± 850 C.01G1W10 Pb: 40.5 ± 9.6	19692 ± 730 C.5W10 Pb: 36.1 ± 5.7	20074 ± 1188 P0A.1B.9W10 Pb: 38.7 ± 12.9
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	20942 ± 628 ★ C.1D.9W50 Pb: 53.7 ± 6.5	20234 ± 615 C.5D.25W50 Pb: 47.4 ± 9.0	19917 ± 515 C1W500 Pb: 48.3 ± 3.4	20199 ± 657 C.5W50 Pb: 47.0 ± 9.1

(a) Results on the Boolean scenario, for ΔT=500 (Naive TCR: 15000, Optimal TCR: 25000)

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	90844 ± 2083 ▲ C5W10 Pb: 69.6 ± 8.2	92361 ± 818 C5G100W1 Pb: 78.0 ± 1.6	91954 ± 1044 ▲ C10W1 Pb: 79.0 ± 2.3	85949 ± 4444 P0A.3B.6W1 Pb: 48.5 ± 16.1
NormIns	90844 ± 2083 ▲ C.5W10 Pb: 69.6 ± 8.2	**93735 ± 1291★** **C.5G10W1** **Pb: 80.4 ± 4.0**	91986 ± 1040 ★ C1W1 Pb: 79.0 ± 2.9	88921 ± 3679 ★ P0A.1B.9W1 Pb: 57.5 ± 12.6
AbsAvg	91285 ± 2300 ★ C5W10 Pb: 70.6 ± 8.2	92650 ± 858 C5G100W10 Pb: 78.8 ± 2.2	91479 ± 1037 ▲ C10W10 Pb: 78.4 ± 2.4	85488 ± 812 P.05A.9B.6W50 Pb: 62.6 ± 3.1
NormAvg	85793 ± 3017 C1W10 Pb: 50.6 ± 11.6	91755 ± 2273 C.01G1W10 Pb: 69.4 ± 8.9	87394 ± 2652 C1W50 Pb: 54.3 ± 7.8	85803 ± 4244 P0A.3B.9W10 Pb: 48.4 ± 13.6
AbsExt	85480 ± 3093 C10W10 Pb: 48.8 ± 12.7	90258 ± 2535 C.01G10W10 Pb: 64.1 ± 9.8	83560 ± 3862 C5W10 Pb: 44.5 ± 7.3	84695 ± 3298 P0A.9B.9W10 Pb: 45.4 ± 10.5
NormExt	85468 ± 3070 C1W10 Pb: 48.9 ± 12.8	90708 ± 2028 C.01G1W10 Pb: 65.6 ± 7.5	83620 ± 3806 C.5W10 Pb: 44.6 ± 7.8	85357 ± 4002 P0A.3B.9W10 Pb: 48.0 ± 13.8
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	90530 ± 1293 ★ C.1D1W50 Pb: 72.2 ± 4.8	87203 ± 1784 C.5D.75W50 Pb: 66.0 ± 7.8	82875 ± 1501 C.1W100 Pb: 53.1 ± 4.1	87108 ± 1576 C.5W50 Pb: 65.7 ± 6.4

(b) Results on the Boolean scenario, for ΔT=2000 (Naive TCR: 60000, Optimal TCR: 100000)

Table 6.8: Results on the Boolean scenario for $\Delta T \in \{500, 2000\}$



(a) NormIns-DMAB (C.5 γ10 W1)

(b) AbsExt-DMAB (C.01 γ10 W10)

(c) NormIns-MAB (C1 W1)

(d) AbsAvg-SLMAB (C5 W10)

(e) Decay/AUC-RMAB (C.1 D1 W50)

(f) NormIns-AP (P0 A.1 B.9 W1)

Figure 6.4: Behavior of DMAB, MAB, SLMAB, RMAB and AP, combined with their best *Credit Assignment* schemes, on the Boolean scenario with $\Delta T = 2000$. DMAB is the overall winner again, combined with the NormIns *Credit Assignment* scheme: very few restarts are correctly triggered. Conversely, when combined with the AbsExt *Credit Assignment*, it triggers several misplaced restarts, while failing to correctly follow the changes (as does NormIns-AP).

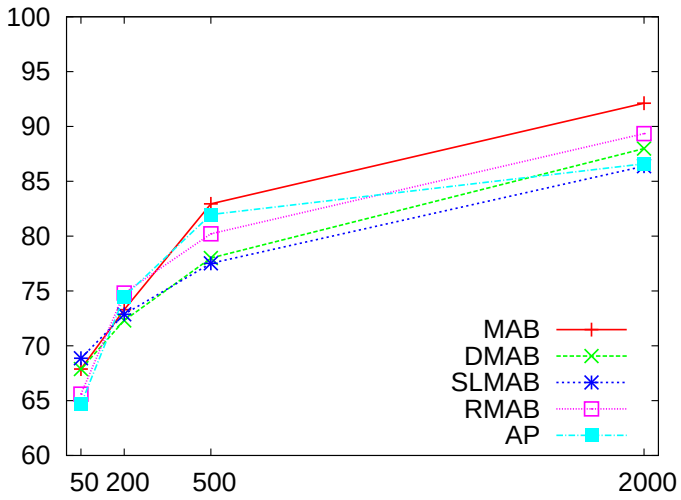
Outlier scenario

The Outlier scenario is by far the most difficult between the three scenarios involving 5 artificial operators. As discussed in Section 5.4.2, although providing very informative rewards about the quality of each operator (such as in the Uniform case, but without any overlapping), the non-zero rewards are very rare (only 10% of the cases), resulting in a huge variance ($V = 225$ for the best operator, while $V = 25$ for the same best in the Boolean scenario, both with $\mathbb{E} = 5$) that greatly complicates the job of the AOS schemes. As for the other two scenarios, empirical results on this scenario are presented in Tables 6.9 and 6.10, respectively, for $\Delta T \in \{50, 200\}$ and $\Delta T \in \{500, 2000\}$.

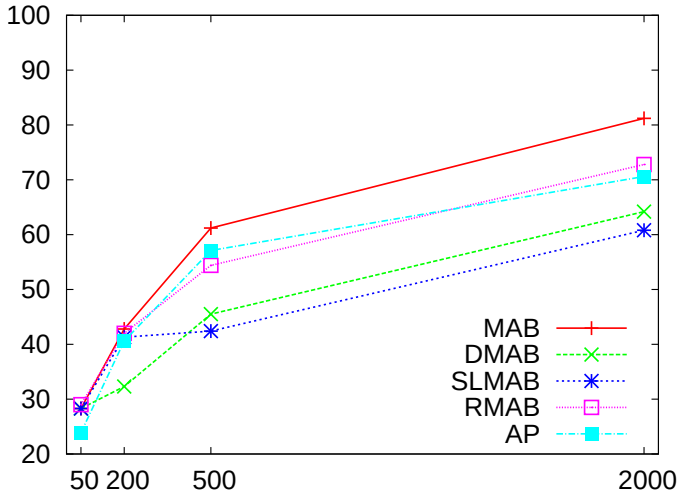
Accordingly, all techniques perform very poorly for small values of ΔT. This is not a surprise, due to the small chance of seeing some outlier reward within 50 or even 200 time steps. For instance, the best TCR obtained over all techniques is 1722 for $\Delta T = 50$ (with a $p(best)$ of only 28%), while the naive approach would do 1500; and 7560 for $\Delta T = 200$, versus 6000 for the naive strategy. However, the situation changes for some techniques when the steady-state period between each change in the rewards distribution is longer (*i.e.*, bigger ΔT, hence more chances of receiving informative rewards), as shown by the scaling of their performances with respect to ΔT in Figure 6.5.

For $\Delta T = 2000$ (Table 6.10b), MAB and RMAB attain, respectively, 92% and 89% of the maximum TCR, with a significant advantage to MAB with respect to RMAB and to all the other techniques: the standard MAB is able of efficiently recognizing and exploiting the best operator (Figure 6.6c). For $\Delta T = 500$, however, RMAB and AP are statistically equivalent to MAB, again winner. Concerning DMAB, differently from the former two artificial scenarios, in this case its restarting mechanism is not able to provide good performance: given the high variance of the rewards received, it is very difficult to find a good value for the Page-Hinkley change-detection threshold γ, and this results in the triggering of several misplaced rewards, as shown in Figures 6.6a and 6.6b. The DMAB is only able to outperform SLMAB, in terms of both TCR and $p(best)$ measures, for the two longest epochs, and AP (only with respect to TCR) for the longest epoch. While AP has its performance hindered again by p_{min}=0.05 (Figure 6.6f), SLMAB fails to efficiently recognize the best operator in the first epoch, takes a long time to adapt to new situations and, finally, it is not able to exploit the best operator more than 80% of the times even at the end of a steady-state epoch as long as 2000 time steps (Figure 6.6d).

Regarding the *Credit Assignment* schemes, here, the Absolute Extreme is clearly the best, as could be expected: when an outlier value is triggered, this scheme will maintain this operator in some top position longer than any other scheme. Interestingly, the Instantaneous reward is a complete disaster for AP, while maintaining a fair level of performance (at least similar to that of the Average reward) for the bandit-based techniques: some average actually takes place in the computation of $\hat{q}$ on the bandit-based approaches (see Equation 5.10), hence keeping some memory of the outlier value; while the benefit of such value vanishes more rapidly within the two-tiered mechanism of AP. It is also clear that Normalization does not work at all here, as it impacts on very few time steps (most rewards are 0), while hiding the outlier effect by bringing the extreme value of the reward back to 1. On the other hand, the rank-based *Credit Assignment* schemes show



(a) % max TCR w.r.t. epoch length



(b) p(best) w.r.t. epoch length

Figure 6.5: Scaling of mean performance (TCR above, and $p(best)$ below) in relation to the epoch length ΔT, for each *Operator Selection* technique with its best *Credit Assignment* scheme, on the Outlier scenario.

their value in this scenario: all the four different variants are statistically equivalent, and also equivalent with respect to the global best method in most epoch lengths; the only exception is the longest epoch, in which the different combinations with RMAB (Figure 6.6e) are still ranked second, but with a significant difference in relation to the best (the AbsExt-MAB). Indeed, as discussed in Section 5.2.4, the use of the rank-based schemes with decay factor tries to mimic, in a smoother way, the intuition of the Extreme *Credit Assignment*; this explains their good performance on this scenario.

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	1633 ± 284 ▲ C5W10 Pb: 25.5 ± 8.0	1673 ± 244 ▲ C10G100W1 Pb: 25.1 ± 5.9	1649 ± 283 ▲ C10W1 Pb: 26.1 ± 5.8	1601 ± 249 ▲ P.05A.1B.9W1 Pb: 21.4 ± 5.7
NormIns	1541 ± 254 C1W100 Pb: 20.5 ± 3.6	1541 ± 254 ▲ C.1G10W1 Pb: 20.1 ± 5.6	1541 ± 254 ▲ C.1W1 Pb: 20.1 ± 5.6	1545 ± 255 ▲ P0A.3B.9W1 Pb: 21.1 ± 8.4
AbsAvg	1677 ± 264 ▲ C10W10 Pb: 26.8 ± 7.3	1650 ± 286 ▲ C5G10W10 Pb: 23.7 ± 5.0	1647 ± 264 ▲ C10W10 Pb: 24.7 ± 7.0	1607 ± 223 ▲ P.1A.6B.3W10 Pb: 22.8 ± 5.2
NormAvg	1586 ± 217 ▲ C5W10 Pb: 22.0 ± 3.4	1591 ± 205 ▲ C1G1W10 Pb: 23.4 ± 4.6	1561 ± 284 ▲ C1W500 Pb: 21.8 ± 5.3	1603 ± 278 ▲ P.05A.9B.9W500 Pb: 21.7 ± 4.4
AbsExt	**1722 ± 236 ★** **C100W10** **Pb: 28.3 ± 5.9**	1697 ± 255 ★ C100G1000W10 Pb: 28.3 ± 6.2	1697 ± 255 ★ C100W10 Pb: 28.3 ± 6.2	1617 ± 232 ★ P.1A.6B.3W10 Pb: 23.9 ± 5.1
NormExt	1575 ± 269 ▲ C5W10 Pb: 22.1 ± 4.2	1568 ± 336 ▲ C.01G1W10 Pb: 21.8 ± 7.0	1550 ± 236 ▲ C5W10 Pb: 22.1 ± 2.1	1616 ± 244 ▲ P.1A.9B.3W10 Pb: 23.6 ± 5.2
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	1634 ± 273 ▲ C1D.75W50 Pb: 26.6 ± 6.5	1640 ± 244 ★ C5D.25W500 Pb: 29.0 ± 4.0	1599 ± 255 ▲ C1W50 Pb: 24.7 ± 6.4	1619 ± 229 ▲ C1W50 Pb: 25.6 ± 5.8

(a) Results on the Outlier scenario, for ΔT=50 (Naive TCR: 1500, Optimal TCR: 2500)

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	7210 ± 650 ▲ C10W50 Pb: 33.8 ± 9.5	7085 ± 643 ▲ C10G1000W1 Pb: 29.7 ± 7.2	7085 ± 643 ▲ C10W1 Pb: 29.7 ± 7.2	6758 ± 599 P.05A.1B.6W1 Pb: 26.0 ± 6.3
NormIns	6150 ± 685 C.1W10 Pb: 20.8 ± 5.5	6090 ± 1003 C.01G1W1 Pb: 19.4 ± 6.1	6082 ± 1031 C.01W1 Pb: 19.4 ± 6.4	6203 ± 667 P.05A.1B.1W1 Pb: 21.8 ± 5.2
AbsAvg	7270 ± 707 ▲ C10W50 Pb: 38.3 ± 10.0	7231 ± 606 ★ C10G1000W10 Pb: 32.3 ± 6.5	7231 ± 606 ▲ C10W10 Pb: 32.3 ± 6.5	6896 ± 730 P.05A.9B.9W50 Pb: 29.1 ± 8.3
NormAvg	6419 ± 673 C5W500 Pb: 24.3 ± 6.9	6710 ± 659 C1G1W50 Pb: 24.7 ± 6.4	6656 ± 803 C1W100 Pb: 24.9 ± 8.0	6881 ± 604 P.05A.6B.9W50 Pb: 28.1 ± 8.3
AbsExt	7288 ± 662 ★ C100W50 Pb: 41.3 ± 8.1	7170 ± 527 ▲ C100G1000W10 Pb: 33.5 ± 5.5	7329 ± 466 ★ C100W50 Pb: 42.8 ± 4.0	7449 ± 641 ★ P.05A.9B.6W50 Pb: 40.6 ± 7.5
NormExt	6801 ± 779 ▲ C1W50 Pb: 27.7 ± 11.8	7086 ± 955 ▲ C1G100W50 Pb: 30.3 ± 8.5	7086 ± 955 ▲ C1W50 Pb: 30.3 ± 8.5	7429 ± 608 ▲ P.05A.9B.9W50 Pb: 39.5 ± 7.1
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	**7481 ± 633 ★** **C1D.75W100** **Pb: 42.0 ± 7.0**	7223 ± 625 ▲ C1D.5W100 Pb: 37.5 ± 7.3	7190 ± 521 ▲ C1W100 Pb: 37.2 ± 6.1	7275 ± 586 ▲ C1W100 Pb: 38.3 ± 6.6

(b) Results on the Outlier scenario, for ΔT=200 (Naive TCR: 6000, Optimal TCR: 10000)

Table 6.9: Results on the Outlier scenario for $\Delta T \in \{50, 200\}$

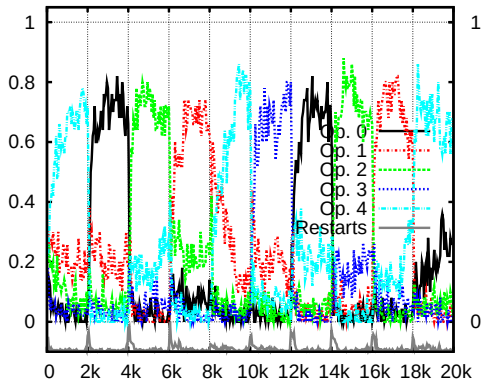
Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	19379 ± 1274 ★ C10W100 Pb: 42.4 ± 9.8	18896 ± 1270 ▲ C10G1000W1 Pb: 37.3 ± 7.5	18920 ± 1279 C10W1 Pb: 37.5 ± 7.7	17391 ± 1145 P.05A.1B.1W1 Pb: 29.1 ± 4.6
NormIns	15418 ± 1705 C.1W500 Pb: 21.6 ± 9.1	15289 ± 1892 C.1G1W1 Pb: 21.0 ± 5.8	15135 ± 1851 C.1W1 Pb: 20.4 ± 5.6	15445 ± 829 P.1A.1B.1W1 Pb: 21.5 ± 2.1
AbsAvg	19156 ± 1227 ▲ C5W50 Pb: 37.3 ± 10.7	19227 ± 1192 ▲ C10G1000W10 Pb: 40.4 ± 8.3	19222 ± 1194 C10W10 Pb: 40.5 ± 8.4	18451 ± 1199 P.05A.9B.6W50 Pb: 34.4 ± 6.6
NormAvg	17077 ± 1909 C1W50 Pb: 26.7 ± 10.4	17950 ± 1327 C1G1W50 Pb: 31.7 ± 6.5	17665 ± 1607 C1W100 Pb: 28.3 ± 7.7	18500 ± 996 P.05A.9B.1W50 Pb: 35.0 ± 6.3
AbsExt	19191 ± 1164 ▲ C100W50 Pb: 45.0 ± 6.5	19461 ± 1042 ▲ C100G1000W50 Pb: 49.4 ± 3.3	**20734 ± 1052★** **C100W50** **Pb: 61.2 ± 4.6**	20491 ± 1128 ★ P.05A.9B.9W50 Pb: 57.1 ± 5.4
NormExt	18130 ± 1990 C1W50 Pb: 29.1 ± 11.7	19500 ± 1227 ★ C1G.1W50 Pb: 45.5 ± 7.7	19038 ± 1747 C1W50 Pb: 36.1 ± 11.3	20362 ± 1151 ▲ P.05A.9B.9W50 Pb: 55.6 ± 5.6
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	19897 ± 982 ▲ C1D.25W100 Pb: 53.0 ± 5.9	20010 ± 993 ▲ C1D.5W100 Pb: 53.4 ± 5.6	20053 ± 981 ★ C1W100 Pb: 54.4 ± 6.4	20012 ± 1095 ▲ C1W100 Pb: 53.4 ± 4.9

(a) Results on the Outlier scenario, for ΔT=500 (Naive TCR: 15000, Optimal TCR: 25000)

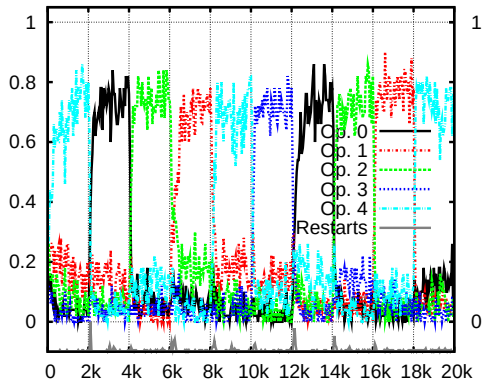
Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	79428 ± 3113 C5W50 Pb: 44.6 ± 7.2	82658 ± 2319 C10G1000W1 Pb: 51.4 ± 7.5	81842 ± 2705 C10W1 Pb: 50.4 ± 7.8	69735 ± 2054 P.05A.1B.6W1 Pb: 29.5 ± 2.5
NormIns	62286 ± 6064 C.5W500 Pb: 21.0 ± 9.1	60675 ± 6308 C.1G1W1 Pb: 20.3 ± 6.7	60298 ± 6874 C.1W1 Pb: 19.8 ± 6.8	61442 ± 2398 P.05A.1B.1W1 Pb: 20.9 ± 2.1
AbsAvg	80723 ± 3381 C5W50 Pb: 41.7 ± 10.6	83515 ± 2814 C5G100W100 Pb: 53.9 ± 7.4	81949 ± 2759 C10W10 Pb: 51.3 ± 7.7	79059 ± 2394 P.05A.6B.1W100 Pb: 44.5 ± 5.7
NormAvg	72584 ± 3463 C5W500 Pb: 37.9 ± 7.6	80043 ± 3641 C1G1W100 Pb: 44.0 ± 8.0	78659 ± 2492 C5W100 Pb: 44.5 ± 5.3	79026 ± 2363 P.05A.6B.1W100 Pb: 44.1 ± 5.8
AbsExt	86372 ± 2602 ★ C100W50 Pb: 60.8 ± 9.1	87699 ± 2108 ▲ C100G1000W50 Pb: 68.4 ± 2.6	**92119 ± 1982★** **C100W50** **Pb: 81.2 ± 2.4**	86595 ± 2035 ★ P.05A.9B.1W50 Pb: 70.6 ± 3.1
NormExt	75437 ± 1787 C5W50 Pb: 37.9 ± 1.7	87978 ± 3308 ★ C1G.1W50 Pb: 64.2 ± 6.8	85638 ± 5590 C1W100 Pb: 53.9 ± 12.3	86474 ± 2039 ▲ P.05A.9B.1W50 Pb: 70.1 ± 3.0
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	89348 ± 2506 ★ C.5D.75W100 Pb: 72.8 ± 4.4	88547 ± 2801 ▲ C.5D.5W100 Pb: 70.2 ± 6.9	89137 ± 3095 ▲ C.5W100 Pb: 71.8 ± 6.8	88785 ± 3214 ▲ C.5W100 Pb: 70.8 ± 7.5

(b) Results on the Outlier scenario, for ΔT=2000 (Naive TCR: 60000, Optimal TCR: 100000)

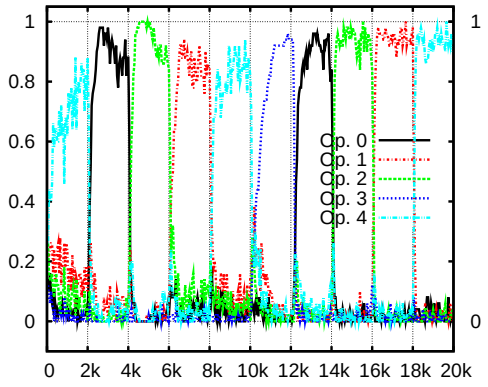
Table 6.10: Results on the Outlier scenario for $\Delta T \in \{500, 2000\}$



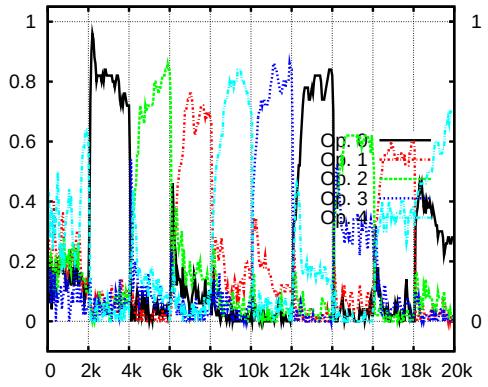
(a) NormExt-DMAB (C1 γ.1 W50)



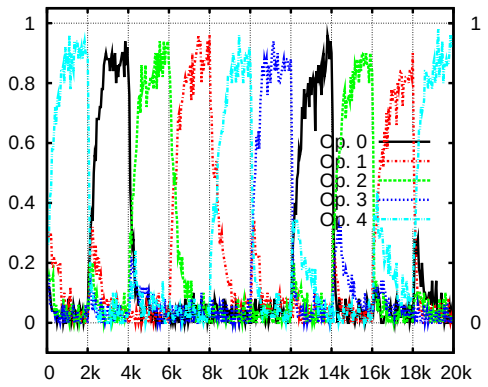
(b) AbsExt-DMAB (C100 γ1000 W50)



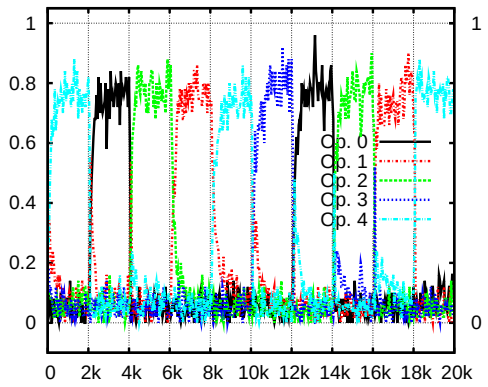
(c) AbsExt-MAB (C100 W50)



(d) AbsExt-SLMAB (C100 W50)



(e) Decay/AUC-RMAB (C.5 D.75 W100)



(f) AbsExt-AP (P.05 A.9 B.1 W50)

Figure 6.6: Behavior of DMAB, MAB, SLMAB, RMAB and AP, combined with their best *Credit Assignment* schemes, on the Outlier scenario with $\Delta T = 2000$. Here, given the high variance of the rewards, the restarts are not helpful: the overall winner is AbsExt-MAB, followed by the robust Decay/AUC-RMAB.

6.3.3 Results on $\mathcal{ART}$ Scenarios

As described in Section 5.4.3, $\mathcal{ART}$ scenarios take into account 2 operators with rewards coming from 2 different $\mathcal{TV}$ distributions. It is important to remember that each distribution is defined by two parameters: the reward R, and the probability p of getting reward R (reward $r = 1$ otherwise); the resulting instance is referred to as $\mathcal{ART}(p_1, R_1, p_2, R_2)$.

While many $\mathcal{ART}$ scenarios with different levels of difficulty were investigated, only the two most representative ones will be considered in the following. It is important to highlight that, for both instances, the two longest epochs ($\Delta T \in \{500, 2000\}$) are used to check how fast each AOS scheme can adapt to a new situation after a long period of stability; hence, only one permutation of rewards is done ($01 \mapsto 10$) in these cases, *i.e.*, only two epochs of length ΔT are considered.

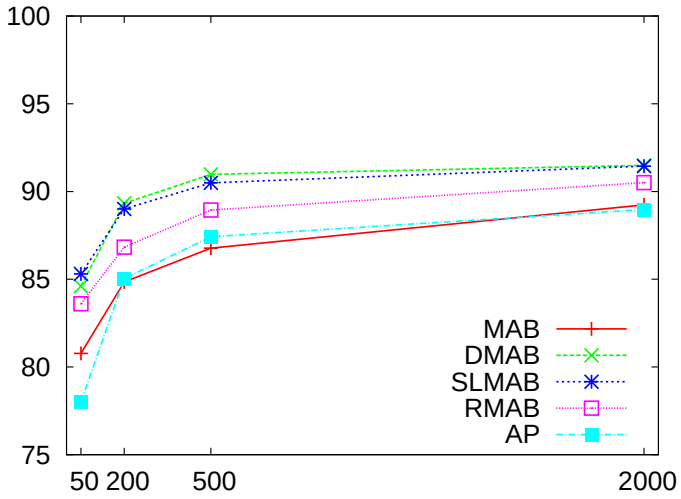
Low Average/High Variance vs. High Average/Low Variance scenario

The $\mathcal{ART}(0.01, 101, 0.5, 10)$ problem involves a low average/high variance distribution ($\mathbb{E}_1 = 2$, $V_1 = 99$) and a high average/low variance distribution ($\mathbb{E}_2 = 6$, $V_2 = 20.25$) operators. Detailed results are presented in Tables 6.11 and 6.12, respectively, for $\Delta T \in \{50, 200\}$, and for $\Delta T \in \{500, 2000\}$.

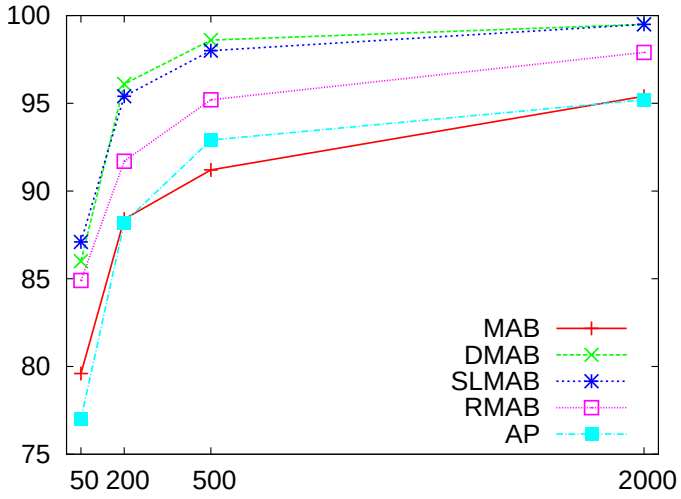
Indeed, the fact that the high-variance operator is also the one with lower reward expectation should make this operator to be easily discarded. Given this clearness in the reward distribution, the Absolute version of all the three kinds of *Credit Assignment* based on the raw values of fitness improvements are able to achieve good performance for all epoch lengths, with higher TCR values being attained by *Operator Selection* techniques employing the Absolute Extreme (AbsExt) *Credit Assignment*. Accordingly, the rank-based AUC *Credit Assignment* schemes also perform well, significantly better than the SR-based variants, with a small (although significant in most cases) difference between its linear (Decay with $D = 1$) and NDCG (equivalent to Decay with $D = 0.4$) variants. This confirms again the fact that, when there are only few possible reward values, the decay factor does not matter much in the differing between the qualities of the operators.

Figure 6.7 shows how the performance of each *Operator Selection* technique, with its corresponding best *Credit Assignment* scheme, scales with respect to the epoch length ΔT. As can be seen, the performance ranking of the *Operator Selection* techniques is very clear, in terms of both TCR and $p(best)$ measures. The overall winner is again DMAB, combined with the AbsExt *Credit Assignment*, which precisely performs restarts every time a change occurs (Figures 6.8a and 6.8b), supported by its well-tuned PH change-detection test. Although having one hyper-parameter less, the SLMAB performs equivalently to the winner DMAB in all cases, with its sliding update rule quickly adapting to the new situation (Figures 6.8c and 6.8d). RMAB comes in third place, combined with AUC with linear decay in all cases, one more time confirming that the adaptation being done on the *Credit Assignment* side is rather efficient (Figures 6.8e and 6.8f). It significantly outperforms the MAB and the AP methods for all epoch lengths, except for the longest epoch, in which the best configuration for AP becomes statistically equivalent to all techniques due to the very high standard deviation on its TCR performance (though with a lower

mean). The standard MAB and the baseline AP *Operator Selection* techniques succeed in following the dynamics of the scenario, but their performances are greatly affected by the facts that their adaptation is slower than those of the others, and they are not able to exploit the best operator up to the maximal rate. By analyzing the behavior plots, it is also interesting to see the gain, in terms of speed of adaptation and exploitation efficiency, provided by the different bandit-based extensions with respect to the original MAB technique (Figures 6.8g and 6.8h).



(a) % max TCR w.r.t. epoch length



(b) p(best) w.r.t. epoch length

Figure 6.7: Scaling of mean performance (TCR above, and $p(best)$ below) in relation to the epoch length ΔT, for each *Operator Selection* technique with its best *Credit Assignment* scheme, on the $\mathcal{ART}(0.01, 101, 0.5, 10)$ scenario.

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	**2559 ± 137 ★** **C5W10** **Pb: 12.9 ± 3.8**	2461 ± 140 ▲ C5G10W1 Pb: 18.8 ± 2.3	2423 ± 142 ★ C10W1 Pb: 20.4 ± 4.6	2341 ± 164 ★ P.1A.9B.9W1 Pb: 23.0 ± 5.3
NormIns	2148 ± 145 C5W100 Pb: 36.2 ± 2.6	2401 ± 162 C1G10W1 Pb: 22.0 ± 3.0	2401 ± 162 ▲ C1W1 Pb: 22.0 ± 3.0	2246 ± 194 ▲ P.1A.9B.9W1 Pb: 27.8 ± 6.2
AbsAvg	2513 ± 131 ▲ C5W10 Pb: 15.4 ± 4.5	2447 ± 192 ▲ C1G10W10 Pb: 19.2 ± 7.0	2304 ± 133 C10W10 Pb: 27.0 ± 4.9	2277 ± 156 ▲ P.1A.9B.9W10 Pb: 26.2 ± 5.3
NormAvg	2139 ± 251 C1W50 Pb: 36.0 ± 10.0	2348 ± 156 C1G10W10 Pb: 25.1 ± 3.5	2348 ± 156 ▲ C1W10 Pb: 25.1 ± 3.5	2186 ± 168 P.1A.9B.9W10 Pb: 31.1 ± 6.2
AbsExt	2465 ± 114 ▲ C10W10 Pb: 18.0 ± 5.8	2538 ± 148 ★ C1G1W10 Pb: 14.0 ± 6.5	2234 ± 205 C10W10 Pb: 30.1 ± 12.7	2276 ± 149 ▲ P.1A.6B.9W10 Pb: 26.3 ± 4.5
NormExt	2141 ± 192 C5W10 Pb: 35.1 ± 4.7	2335 ± 149 C1G10W10 Pb: 25.4 ± 4.8	2332 ± 149 ▲ C1W10 Pb: 25.4 ± 4.8	2207 ± 166 P.1A.6B.9W10 Pb: 29.9 ± 5.8
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	2508 ± 136 ★ C.01D1W10 Pb: 15.1 ± 1.6	2371 ± 153 C1D.9W10 Pb: 23.3 ± 1.3	2499 ± 129 ▲ C.01W10 Pb: 15.3 ± 1.9	2365 ± 131 C1W10 Pb: 22.7 ± 1.5

(a) Results $\mathcal{ART}(0.01, 101, 0.5, 10)$ scenario, for $\Delta T = 50$ (Naive TCR: 2000, Optimal TCR: 3000)

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	10656 ± 226 ▲ C5W10 Pb: 5.1 ± 1.4	10348 ± 243 C5G100W1 Pb: 10.1 ± 2.7	10182 ± 243 ★ C10W1 Pb: 11.6 ± 1.9	10204 ± 297 ★ P.05A.6B.9W1 Pb: 11.8 ± 2.8
NormIns	8635 ± 278 C10W500 Pb: 33.4 ± 0.8	10012 ± 273 C1G100W1 Pb: 14.1 ± 1.8	10012 ± 273 C1W1 Pb: 14.1 ± 1.8	9921 ± 296 P.1A.6B.9W1 Pb: 15.7 ± 2.9
AbsAvg	10637 ± 229 ▲ C5W10 Pb: 5.2 ± 1.4	10483 ± 262 C1G10W10 Pb: 7.5 ± 2.7	10063 ± 260 ▲ C10W10 Pb: 13.5 ± 2.3	10159 ± 303 ▲ P.05A.9B.9W10 Pb: 12.3 ± 3.3
NormAvg	8777 ± 260 C5W50 Pb: 32.3 ± 2.4	9899 ± 442 C1G100W10 Pb: 15.7 ± 4.8	9899 ± 442 C1W10 Pb: 15.7 ± 4.8	9875 ± 258 P.1A.9B.9W10 Pb: 16.4 ± 2.4
AbsExt	10680 ± 216 ★ C10W10 Pb: 4.6 ± 1.7	**10719 ± 221 ★** **C5G10W10** **Pb: 3.9 ± 0.4**	9322 ± 1010 C10W10 Pb: 23.9 ± 15.5	10149 ± 300 ▲ P.05A.9B.9W10 Pb: 12.4 ± 3.3
NormExt	8503 ± 278 C5W50 Pb: 35.5 ± 3.6	9778 ± 399 C1G100W10 Pb: 17.5 ± 5.1	9778 ± 399 C1W10 Pb: 17.5 ± 5.1	9893 ± 270 P.1A.6B.9W10 Pb: 16.3 ± 2.5
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	10419 ± 262 ★ C.5D1W100 Pb: 8.3 ± 1.5	10052 ± 280 C.5D.9W10 Pb: 12.8 ± 1.4	10160 ± 250 C.01W10 Pb: 11.6 ± 0.6	9977 ± 259 C.5W10 Pb: 13.9 ± 2.1

(b) Results $\mathcal{ART}(0.01, 101, 0.5, 10)$ scenario, for $\Delta T = 200$ (Naive TCR: 8000, Optimal TCR: 12000)

Table 6.11: Results on $\mathcal{ART}(0.01, 101, 0.5, 10)$, 10 epochs for $\Delta T \in \{50, 200\}$

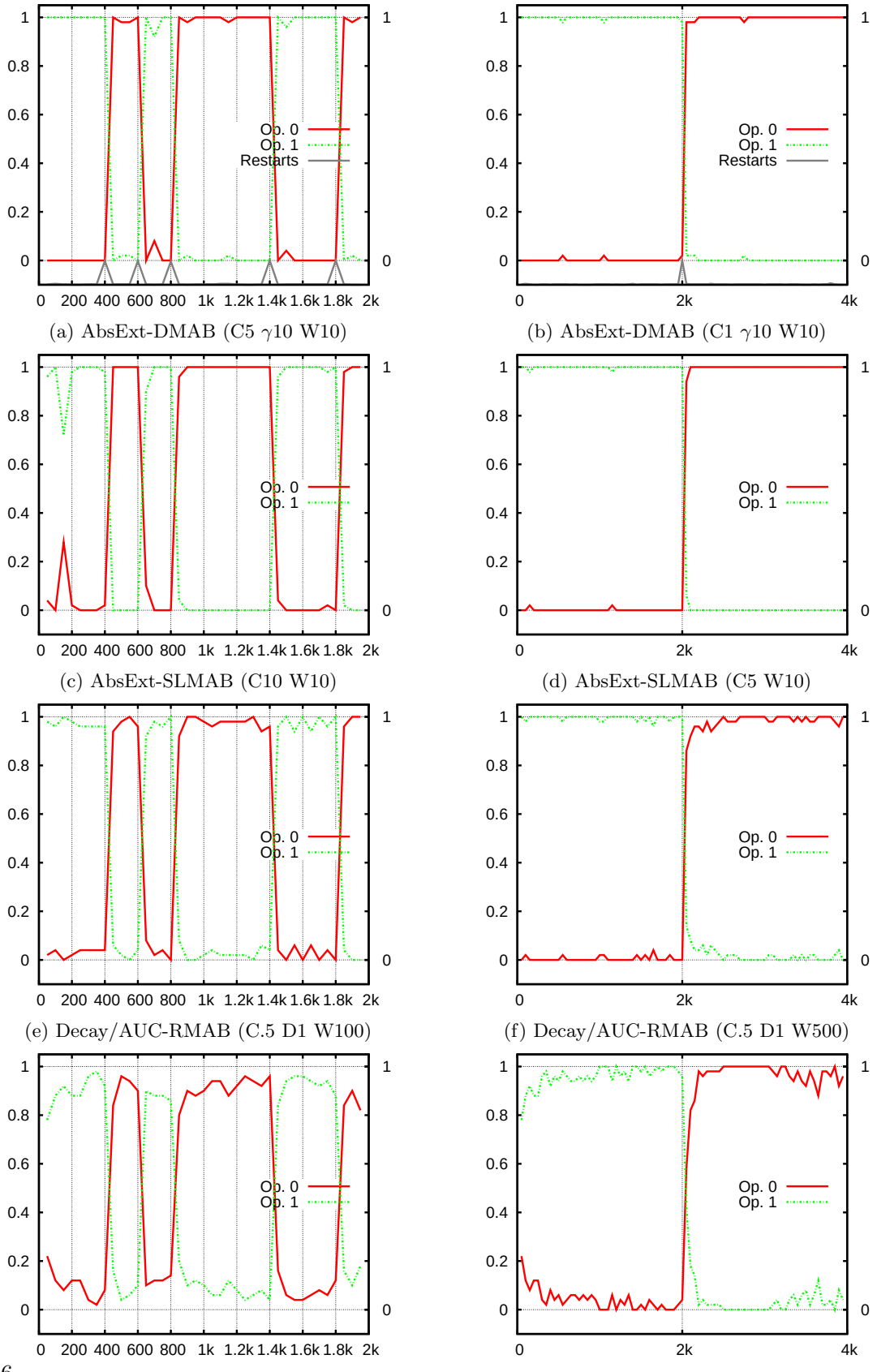
Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	5400 ± 127 ▲ C1W10 Pb: 2.8 ± 1.6	5291 ± 173 C5G100W1 Pb: 6.8 ± 4.1	5190 ± 169 ▲ C5W1 Pb: 9.4 ± 5.0	5246 ± 416 ★ P0A.9B.1W1 Pb: 7.1 ± 12.7
NormIns	5217 ± 157 C5W500 Pb: 8.2 ± 0.5	4923 ± 273 C.5G10W1 Pb: 16.4 ± 6.8	4923 ± 273 C.5W1 Pb: 16.4 ± 6.8	5085 ± 267 P.05A.9B.9W1 Pb: 12.0 ± 6.1
AbsAvg	5377 ± 124 ▲ C1W10 Pb: 3.5 ± 1.7	5334 ± 211 ▲ C1G10W10 Pb: 5.1 ± 4.0	5120 ± 159 ▲ C10W10 Pb: 10.9 ± 3.6	5209 ± 403 ▲ P0A.9B.1W10 Pb: 8.0 ± 12.4
NormAvg	4978 ± 705 C.5W100 Pb: 14.8 ± 20.5	5206 ± 168 C1G10W10 Pb: 8.8 ± 3.2	5206 ± 169 ★ C1W10 Pb: 8.8 ± 3.2	5068 ± 257 P.05A.9B.9W10 Pb: 12.2 ± 6.1
AbsExt	5430 ± 132 ★ C5W10 Pb: 2.0 ± 1.1	**5459 ± 142 ★** **C1G1W10** **Pb: 1.4 ± 0.4**	4828 ± 614 ▲ C10W10 Pb: 19.6 ± 19.5	5206 ± 402 ▲ P0A.9B.1W10 Pb: 8.2 ± 12.4
NormExt	4604 ± 189 C10W10 Pb: 26.1 ± 2.4	5023 ± 338 C1G1W10 Pb: 13.8 ± 8.3	5013 ± 334 ▲ C1W10 Pb: 14.2 ± 8.6	5070 ± 256 P.05A.6B.9W10 Pb: 12.1 ± 6.1
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	5336 ± 158 ★ C.1D1W50 Pb: 4.8 ± 1.4	5115 ± 156 C1D.75W50 Pb: 11.0 ± 1.3	5237 ± 157 ▲ C.01W50 Pb: 7.8 ± 4.4	5069 ± 149 C1W50 Pb: 12.3 ± 2.6

(a) Results $\mathcal{ART}(0.01, 101, 0.5, 10)$ scenario, for $\Delta T = 500$ (Naive TCR: 4000, Optimal TCR: 6000)

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	21926 ± 320 ▲ C1W10 Pb: 0.7 ± 0.4	21683 ± 650 ▲ C1G100W1 Pb: 2.4 ± 3.6	21417 ± 373 ★ C10W1 Pb: 4.6 ± 2.0	21354 ± 1871 ★ P0A.9B.1W1 Pb: 4.8 ± 13.4
NormIns	18714 ± 2678 C.5W500 Pb: 23.3 ± 19.0	19971 ± 449 C1G100W1 Pb: 14.7 ± 1.2	19971 ± 449 C1W1 Pb: 14.7 ± 1.2	21022 ± 353 P.05A.3B.1W1 Pb: 7.3 ± 1.1
AbsAvg	21904 ± 321 ▲ C1W10 Pb: 0.9 ± 0.5	21705 ± 346 C1G10W50 Pb: 2.3 ± 1.0	21344 ± 366 ▲ C10W10 Pb: 4.7 ± 1.6	21322 ± 1861 ▲ P0A.9B.1W10 Pb: 5.0 ± 13.4
NormAvg	20221 ± 3445 C.5W500 Pb: 12.8 ± 24.4	21057 ± 830 C1G.1W10 Pb: 6.8 ± 5.3	20994 ± 849 ▲ C1W10 Pb: 7.2 ± 4.7	21010 ± 326 P.05A.9B.9W10 Pb: 7.3 ± 1.5
AbsExt	21947 ± 326 ★ C5W10 Pb: 0.5 ± 0.4	**21958 ± 360 ★** **C1G10W10** **Pb: 0.5 ± 0.6**	19974 ± 2399 ▲ C10W10 Pb: 14.9 ± 18.7	21317 ± 1860 ▲ P0A.9B.1W10 Pb: 5.1 ± 13.4
NormExt	18490 ± 391 C5W50 Pb: 24.8 ± 3.5	20806 ± 658 C1G.01W10 Pb: 8.6 ± 3.9	20480 ± 1173 C1W10 Pb: 11.0 ± 8.3	21008 ± 326 P.05A.6B.9W10 Pb: 7.3 ± 1.5
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	21722 ± 371 ★ C.5D1W500 Pb: 2.1 ± 1.0	21204 ± 388 C.5D.75W50 Pb: 5.7 ± 0.9	21415 ± 310 C.1W50 Pb: 4.3 ± 1.3	21078 ± 509 C.01W50 Pb: 6.5 ± 3.6

(b) Results $\mathcal{ART}(0.01, 101, 0.5, 10)$ scenario, for $\Delta T = 2000$ (Naive TCR: 16000, Optimal TCR: 24000)

Table 6.12: Results $\mathcal{ART}(0.01, 101, 0.5, 10)$, 2 epochs for $\Delta T \in \{500, 2000\}$



(a) AbsExt-DMAB (C5 γ10 W10)

(b) AbsExt-DMAB (C1 γ10 W10)

(c) AbsExt-SLMAB (C10 W10)

(d) AbsExt-SLMAB (C5 W10)

(e) Decay/AUC-RMAB (C.5 D1 W100)

(f) Decay/AUC-RMAB (C.5 D1 W500)

(g) AbsIns-MAB (C10 W1)

(h) AbsIns-MAB (C10 W1)

Figure 6.8: Behavior of DMAB, SLMAB, RMAB and MAB, combined with their best *Credit Assignment* schemes, on the $\mathcal{ART}(0.01, 101, 0.5, 10)$ scenario, for $\Delta T = 200$ on the left column, and $\Delta T = 2000$ on the right column.

High Average/High Variance vs. Low Average/Low Variance scenario

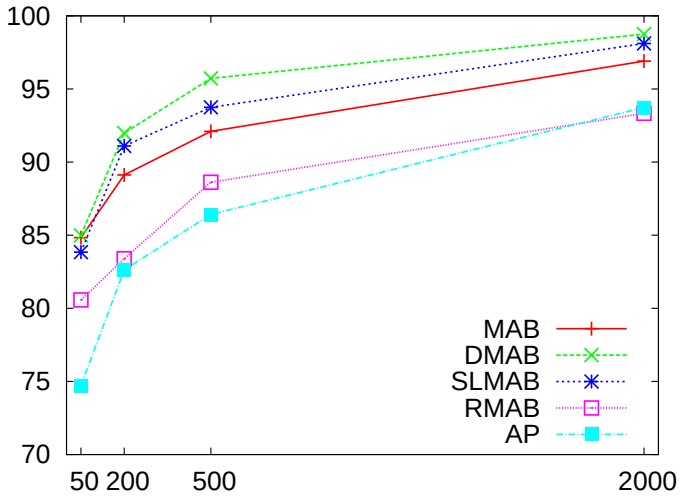
Oppositely to the previous $\mathcal{ART}$ scenario, the $\mathcal{ART}(0.1, 39, 0.5, 3)$ problem has a high average/high variance ($\mathbb{E}_1 = 4.8$, $V_1 = 130$) and a low average/low variance ($\mathbb{E}_2 = 2$ and $V_2 = 1$) operators. Detailed results are depicted in Tables 6.13 and 6.14, respectively, for $\Delta T \in \{50, 200\}$ and $\Delta T \in \{500, 2000\}$.

This scenario is much more complex than the previous one, as the regularity of the second operator might lead the AOS schemes to believe it is the best operator, while in fact the best is the first one. This kind of situation was the main motivation for the proposal of the Extreme-based *Credit Assignment*, as discussed in Section 5.2.2, which indeed performs significantly better than all the other schemes for the three longest epoch lengths, while also being the winner but statistically equivalent to several others for $\Delta T = 50$. As expected, the Instantaneous schemes achieve a much lower performance: as they assign credit based on a single operator application, they need to be very lucky in order to catch the outlier reward that is received only in 10% of the cases for the first operator. This situation is alleviated by the Average *Credit Assignment*, but not enough to provide good performance. As in the Outlier scenario, the Normalized variants of these schemes do not work at all: the very different reward values of 3 and 39 given by each of the operators result in the same credit value of 1 in different moments of the search, as discussed in Section 5.2.4.

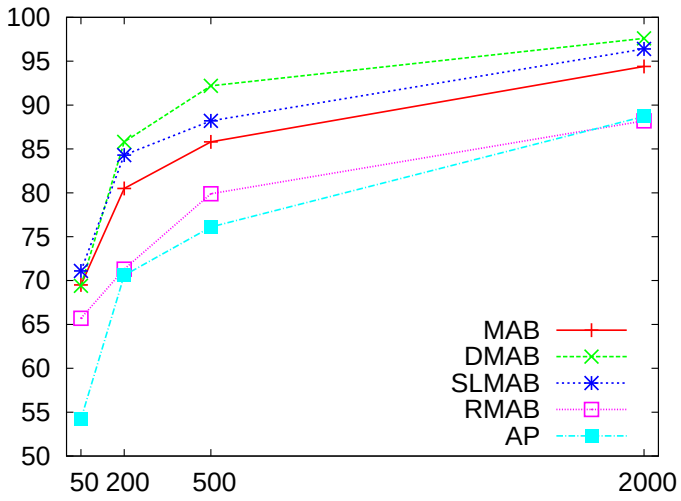
Surprisingly, the different AOS combinations involving the rank-based *Credit Assignment* schemes with the RMAB *Operator Selection* technique do not work well at all in this case, even when employing a strong decaying factor (what intuitively approaches it to the behavior of the Extreme *Credit Assignment* scheme, as discussed in Section 5.2.4). A tentative interpretation, based on the high variation of the instant selection rates depicted in Figures 6.10e and 6.10f, is developed as follows. In the Extreme *Credit Assignment*, when the outlier operator receives the high reward ($R = 39$ in this case), a equivalent credit of 39 is assigned to this operator at least W times, no matter how many bad rewards it receives during this period, and no matter how many times the other (regular) operator is applied, as there is a separate window for the rewards received by each operator. In the rank-based schemes, as there is only one sliding window for all operators, this dominance is greatly reduced by two factors: the applications of the other operator, which, besides pushing the outlier reward out of the window, will always reduce the overall quality estimate of the outlier operator; besides, the receiving of bad rewards by the outlier operator (what happens in 90% of the times in this case), will also affect its quality estimate, consequently promoting the exploration of the other operator. And then, given the regularity of the other operator, one trial might be enough to start a long period of dominance.

Anyway, as for the other scenarios, the longer the steady-state epoch, the better the performance for all the techniques (even for the rank-based ones) with respect to the Naive approach, as summarized in Figure 6.9. Notably, both DMAB and SLMAB reach a quasi-perfect score when $\Delta T = 2000$. The DMAB is once more the overall winner for all epoch lengths, but statistically equivalent to the second and third best techniques, namely, the SLMAB and the standard MAB, all of them combined with the AbsExt *Credit Assignment* scheme. The baseline AP is also once more (significantly) beaten by all the

bandit-based approaches, with the exception of the RMAB, which performs equally bad, for the reasons previously discussed. The behavior plots of all the winner configurations for the bandit-based approaches are presented in Figure 6.10.



(a) % max TCR w.r.t. epoch length



(b) p(best) w.r.t. epoch length

Figure 6.9: Scaling of mean performance (TCR above, and $p(best)$ below) in relation to the epoch length ΔT, for each *Operator Selection* technique with its best *Credit Assignment* scheme, on the $\mathcal{ART}(0.1, 39, 0.5, 3)$ scenario.

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	1843 ± 272 ▲ C10W10 Pb: 60.1 ± 7.1	1864 ± 256 C10G100W1 Pb: 60.4 ± 7.3	1862 ± 255 ▲ C10W1 Pb: 58.8 ± 7.8	1737 ± 246 ▲ P.1A.1B.3W1 Pb: 50.0 ± 9.1
NormIns	1720 ± 180 C100W10 Pb: 49.7 ± 1.8	1714 ± 177 C10G.01W1 Pb: 49.7 ± 0.6	1687 ± 157 C100W1 Pb: 49.8 ± 0.2	1718 ± 160 ▲ P0A.9B.1W1 Pb: 51.1 ± 3.4
AbsAvg	1927 ± 269 ▲ C10W10 Pb: 65.2 ± 7.1	1946 ± 247 ▲ C10G100W10 Pb: 63.3 ± 6.2	1894 ± 228 ▲ C10W10 Pb: 61.8 ± 6.5	1761 ± 229 ▲ P.1A.6B.6W50 Pb: 53.0 ± 8.0
NormAvg	1782 ± 202 C5W50 Pb: 54.4 ± 3.3	1757 ± 194 C10G100W10 Pb: 51.8 ± 1.6	1757 ± 194 C10W10 Pb: 51.8 ± 1.6	1768 ± 213 ▲ P.1A.9B.6W50 Pb: 53.0 ± 8.1
AbsExt	2012 ± 301 ★ C100W10 Pb: 71.1 ± 5.9	**2040 ± 238 ★** **C100G1000W10** **Pb: 69.4 ± 4.5**	2036 ± 239 ★ C100W10 Pb: 69.5 ± 4.5	1793 ± 252 ★ P.1A.1B.6W10 Pb: 54.2 ± 9.1
NormExt	1772 ± 196 C5W50 Pb: 55.1 ± 5.2	1758 ± 218 C5G100W10 Pb: 51.0 ± 3.5	1758 ± 218 C5W10 Pb: 51.0 ± 3.5	1743 ± 240 ▲ P.05A.1B.1W10 Pb: 50.2 ± 9.5
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	1934 ± 245 ★ C5D.9W100 Pb: 65.7 ± 6.8	1860 ± 199 ▲ C10D.5W100 Pb: 59.6 ± 3.0	1844 ± 241 ▲ C5W100 Pb: 60.3 ± 6.8	1869 ± 227 ▲ C5W100 Pb: 61.1 ± 8.3

(a) Results $\mathcal{ART}(0.1, 39, 0.5, 3)$ scenario, for $\Delta T = 50$ (Naive TCR: 1700, Optimal TCR: 2400)

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	8337 ± 537 C10W50 Pb: 77.5 ± 3.8	8232 ± 475 C10G100W1 Pb: 74.2 ± 3.0	7996 ± 572 C10W1 Pb: 70.1 ± 7.2	7277 ± 639 P.2A.1B.1W1 Pb: 59.1 ± 5.0
NormIns	6961 ± 628 C1W50 Pb: 51.7 ± 8.8	6830 ± 361 C.1G.01W1 Pb: 50.4 ± 1.7	6792 ± 340 C.01W1 Pb: 50.1 ± 0.1	6820 ± 379 P0A.9B.6W1 Pb: 50.0 ± 0.1
AbsAvg	8522 ± 497 ▲ C10W50 Pb: 79.4 ± 3.3	8344 ± 538 C10G100W10 Pb: 76.1 ± 3.7	8160 ± 466 C10W10 Pb: 73.0 ± 4.3	7795 ± 741 ▲ P.1A.6B.3W50 Pb: 67.9 ± 7.4
NormAvg	7700 ± 450 C5W100 Pb: 65.6 ± 2.6	8110 ± 581 C1G.1W50 Pb: 72.6 ± 6.0	7871 ± 790 C1W50 Pb: 68.1 ± 9.9	7780 ± 526 ▲ P.2A.9B.6W50 Pb: 67.7 ± 3.9
AbsExt	8746 ± 511 ★ C100W50 Pb: 84.3 ± 2.9	**8830 ± 478 ★** **C10G1W50** **Pb: 85.8 ± 1.6**	8555 ± 530 ★ C100W10 Pb: 80.5 ± 3.3	7931 ± 606 ★ P.1A.3B.6W50 Pb: 70.6 ± 5.3
NormExt	7944 ± 376 C5W50 Pb: 71.1 ± 2.1	8046 ± 634 C1G100W50 Pb: 71.5 ± 8.9	8046 ± 634 C1W50 Pb: 71.5 ± 8.9	7830 ± 486 ▲ P.2A.9B.6W50 Pb: 68.5 ± 2.9
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	8006 ± 667 ★ C1D.9W100 Pb: 71.3 ± 8.3	7887 ± 433 ▲ C5D.25W100 Pb: 67.7 ± 1.4	7848 ± 701 ▲ C1W50 Pb: 67.8 ± 7.6	7819 ± 428 ▲ C5W100 Pb: 66.4 ± 1.7

(b) Results $\mathcal{ART}(0.1, 39, 0.5, 3)$ scenario, for $\Delta T = 200$ (Naive TCR: 6800, Optimal TCR: 9600)

Table 6.13: Results $\mathcal{ART}(0.1, 39, 0.5, 3)$, 10 epochs for $\Delta T \in \{50, 200\}$

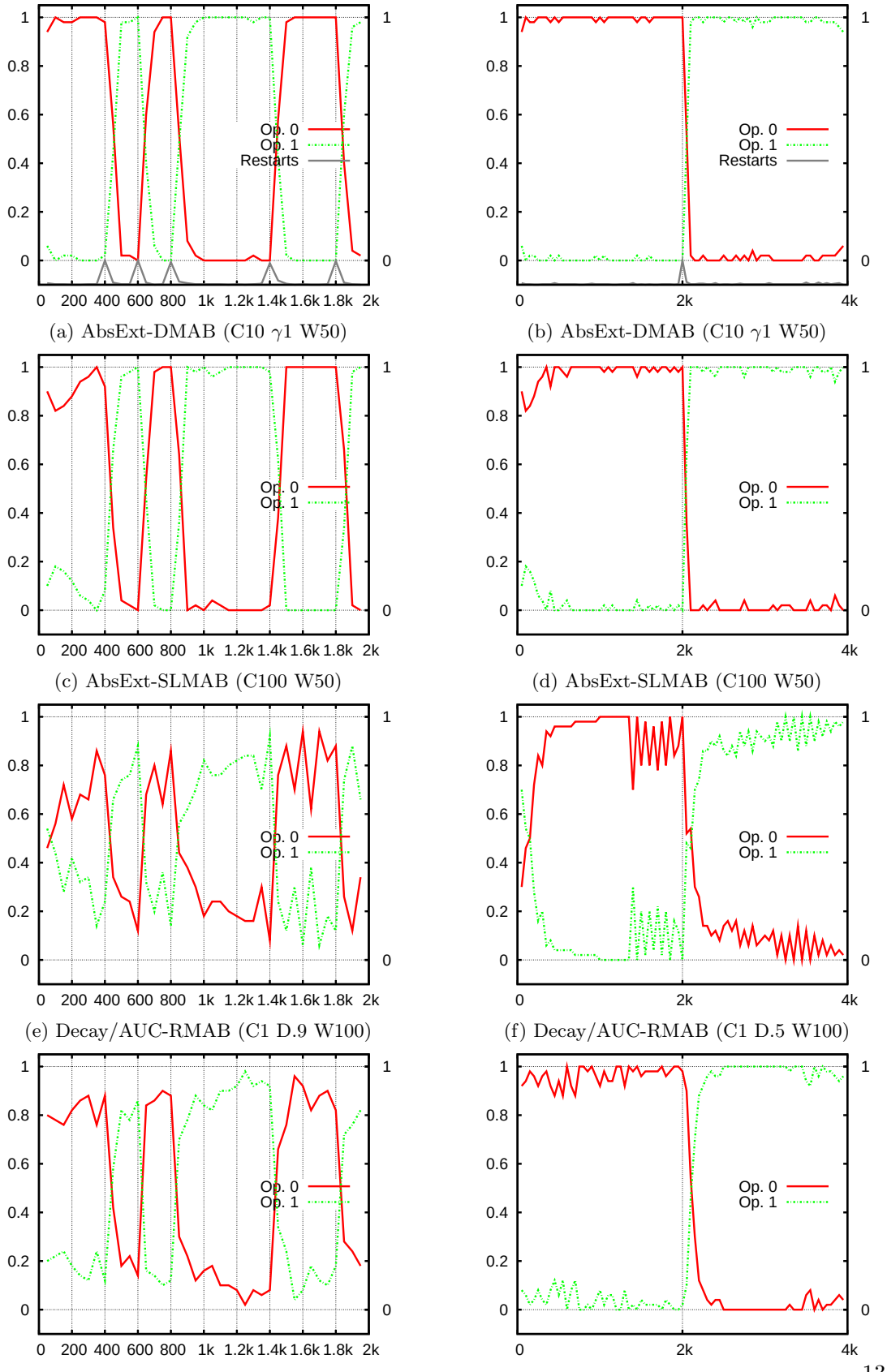
Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	4167 ± 342 C5W50 Pb: 77.0 ± 6.2	4242 ± 421 C5G100W1 Pb: 79.2 ± 8.3	4202 ± 431 C10W1 Pb: 77.2 ± 8.6	3737 ± 402 P.2A.1B.1W1 Pb: 61.6 ± 5.6
NormIns	3661 ± 650 C1W100 Pb: 58.3 ± 22.1	3425 ± 291 C.1G.1W1 Pb: 51.1 ± 6.1	3411 ± 240 C.01W1 Pb: 50.1 ± 0.3	3451 ± 239 P0A.3B.1W1 Pb: 50.8 ± 1.8
AbsAvg	4309 ± 371 ▲ C5W50 Pb: 80.7 ± 6.4	4377 ± 366 ▲ C5G100W10 Pb: 83.6 ± 7.3	4175 ± 400 C10W10 Pb: 76.5 ± 8.1	4116 ± 481 ▲ P.05A.6B.1W100 Pb: 74.7 ± 11.4
NormAvg	3775 ± 280 C5W100 Pb: 63.9 ± 2.1	4241 ± 457 C1G.01W100 Pb: 79.4 ± 11.1	4215 ± 483 ▲ C1W100 Pb: 78.8 ± 11.5	4122 ± 355 ▲ P.1A.9B.1W100 Pb: 75.4 ± 6.6
AbsExt	4500 ± 324 ★ C100W50 Pb: 88.2 ± 2.1	**4595 ± 339 ★** **C10G1W50** **Pb: 92.2 ± 3.6**	4421 ± 297 ★ C100W50 Pb: 85.8 ± 3.5	4148 ± 397 ★ P.1A.1B.3W50 Pb: 76.1 ± 7.8
NormExt	3972 ± 281 C5W100 Pb: 70.2 ± 1.9	4281 ± 444 C1G.01W50 Pb: 80.9 ± 12.1	4203 ± 519 ▲ C1W50 Pb: 78.3 ± 13.2	4146 ± 422 ▲ P.1A.3B.3W50 Pb: 75.9 ± 8.3
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	4078 ± 437 ▲ C1D.25W50 Pb: 74.0 ± 11.3	4254 ± 298 ★ C5D.25W500 Pb: 79.9 ± 3.2	4039 ± 478 ▲ C1W50 Pb: 72.2 ± 12.7	4214 ± 293 ▲ C5W500 Pb: 79.5 ± 3.4

(a) Results $\mathcal{ART}(0.1, 39, 0.5, 3)$ scenario, for $\Delta T = 500$ (Naive TCR: 3400, Optimal TCR: 4800)

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	17725 ± 835 C5W50 Pb: 86.6 ± 3.6	18006 ± 842 C10G1000W1 Pb: 89.0 ± 4.0	17994 ± 848 C10W1 Pb: 89.0 ± 4.1	15021 ± 860 P.2A.1B.1W1 Pb: 62.5 ± 3.4
NormIns	14622 ± 1443 C5W10 Pb: 60.0 ± 11.6	13650 ± 489 C100G.01W1 Pb: 49.9 ± 0.2	13550 ± 477 C.01W1 Pb: 50.0 ± 0.1	13735 ± 504 P0A.3B.1W1 Pb: 50.2 ± 0.4
AbsAvg	18169 ± 755 C5W50 Pb: 90.4 ± 2.5	18299 ± 883 C5G100W100 Pb: 91.9 ± 3.9	17962 ± 837 C10W10 Pb: 88.7 ± 3.9	17873 ± 1272 ▲ P.05A.9B.1W100 Pb: 87.7 ± 7.1
NormAvg	16185 ± 857 C5W500 Pb: 73.0 ± 4.2	18172 ± 1121 C1G1W100 Pb: 90.6 ± 6.9	18172 ± 1124 ▲ C1W100 Pb: 90.8 ± 5.8	17932 ± 1264 ▲ P.05A.9B.1W100 Pb: 88.1 ± 7.5
AbsExt	18838 ± 726 ★ C100W50 Pb: 96.4 ± 0.8	**18960 ± 750 ★** **C10G1W50** **Pb: 97.6 ± 1.1**	18606 ± 743 ★ C100W50 Pb: 94.4 ± 1.9	17976 ± 1053 ▲ P.05A.1B.1W50 Pb: 88.7 ± 5.4
NormExt	17009 ± 598 C5W100 Pb: 80.7 ± 0.7	17969 ± 665 C5G.01W100 Pb: 88.2 ± 0.7	17841 ± 712 C5W100 Pb: 87.6 ± 0.7	17996 ± 967 ★ P.05A.9B.1W100 Pb: 88.7 ± 4.9
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	17921 ± 822 ★ C1D.5W100 Pb: 88.2 ± 4.1	17866 ± 800 ▲ C1D.25W100 Pb: 87.7 ± 4.0	17911 ± 841 ▲ C1W100 Pb: 88.1 ± 4.1	17795 ± 881 ▲ C1W100 Pb: 86.8 ± 5.0

(b) Results $\mathcal{ART}(0.1, 39, 0.5, 3)$ scenario, for $\Delta T = 2000$ (Naive TCR: 13600, Optimal TCR: 19200)

Table 6.14: Results $\mathcal{ART}(0.1, 39, 0.5, 3)$, 2 epochs for $\Delta T \in \{500, 2000\}$



(a) AbsExt-DMAB (C10 γ1 W50)

(b) AbsExt-DMAB (C10 γ1 W50)

(c) AbsExt-SLMAB (C100 W50)

(d) AbsExt-SLMAB (C100 W50)

(e) Decay/AUC-RMAB (C1 D.9 W100)

(f) Decay/AUC-RMAB (C1 D.5 W100)

(g) AbsExt-MAB (C100 W10)

(h) AbsExt-MAB (C100 W50)

Figure 6.10: Behavior of DMAB, SLMAB, RMAB and MAB, combined with their best *Credit Assignment* schemes, on the $\mathcal{ART}(0.1, 39, 0.5, 3)$ scenario, for $\Delta T = 200$ on the left column, and $\Delta T = 2000$ on the right column.

6.3.4 Discussion

The problems used in this Section provided conditions that are very artificial, with abrupt changes happening every ΔT time steps. In real optimization problems, these abrupt changes in the rewards distribution might also occur (*e.g.*, when escaping from a local optima and reaching a new region of the search space); but globally, the dynamics of the operator qualities tends to be much more complex and usually unpredictable (except for the simple benchmark problems, such as the OneMax; see Section 6.4.2). However, the initial motivation for using these artificial scenarios is confirmed by the experimental findings: they enable the detailed analysis of the behavior of the AOS methods, and the verification of their characteristics in practice. The results presented in this Section can be summarized as follows.

The baseline probability-based AP method, although systematically (and significantly) outperforming the original Probability Matching (PM) method (results for PM are not shown here; see Sections 4.4.1 and 4.4.2 for their description), still provides a slow adaptation when compared to all proposed bandit-based approaches. The main reason for this, in most cases, is the limitation provided by the enforced minimal level of exploration p_{min}. But, even when p_{min} is set to zero, its two-tiered update mechanism needs some time in order to start to efficiently exploit the new best operator.

As for the bandit-based approaches, the standard MAB *Operator Selection* technique is able to follow the dynamics in an efficient way when the changes happen smoothly, *i.e.*, when the magnitude of the adaptation to be done is small (*e.g.*, in the Uniform scenario, when the second best operator becomes the best, or vice-versa). Whenever faster dynamics are considered, the DMAB succeeds in adapting very quickly to new situations, supported by its change-detection mechanism. Moreover, as originally expected, the SLMAB is able to perform as efficiently as the DMAB in most cases, due to its parameterless window-based relaxation mechanism.

The RMAB, using any of the proposed rank-based *Credit Assignment* schemes, outperforms the baseline AP and performs equivalently to the standard MAB. But the main benefit brought by these rank-based schemes (as well as by the Normalized versions of the Instantaneous, Average and Extreme schemes) is a higher robustness with respect to: (i) different values of rewards gathered in different stages of the search; and (ii) different (unknown) fitness ranges provided by different problems. Both issues were not assessed in this experimental set. Conversely, for each scenario, the same range of reward values was considered during all the optimization process; besides, each scenario/epoch length was independently tackled after a preliminary off-line tuning phase, what explains why the very sensitive and problem-dependent (assumed after discussion in Sections 5.2.3 and 5.3.3, and confirmed by the sensitivity analysis that will be presented in Section 6.7) AbsExt-DMAB combination was found to be the winner in almost all the cases.

Indeed, the gain in robustness provided by RMAB results in a gain in performance, in relation to the other AOS methods, when several problems are tackled using the same hyper-parameter setting. This situation will be further stressed in the experiments that will be analyzed in the following, specially in the hyper-parameter sensitivity analysis considering different optimization benchmark problems (Section 6.7.1).

6.4 On Boolean Benchmark Problems

Some EA boolean benchmark problems were also used to empirically compare the AOS schemes, *in situ*, *i.e.*, combined with an EA and selecting between actual evolutionary operators on some (still artificial) fitness landscapes with different complexities and levels of difficulty with respect to AOS. Needless to say, in these cases, the dynamics of the performance of the operators are not deterministically switched after every epoch, but rather depending on the evolution trajectory and the fitness landscape.

Three different problems were considered, namely: the eternal OneMax problem, and two harder problems, the Long K-Path and the Royal Road. The experimental settings, in complement to the general settings presented in Section 6.2, will be described in Section 6.4.1. The problems will be presented, and the empirical results will be analyzed, in Sections 6.4.2, 6.4.3 and 6.4.4, respectively, for the OneMax, Long K-Path, and Royal Road. Finally, Section 6.4.5 will conclude this analysis with a discussion about the highlights of these experiments. The results that will be analyzed here were partially published in [Fialho *et al.*, 2008; Fialho *et al.*, 2009a; Fialho *et al.*, 2010c].

6.4.1 Experimental Settings

The performance of the AOS schemes embedded within real EAs is assessed by the number of generations needed to achieve the optimal solution, the lower the better. The resulting total number of fitness evaluations can be roughly measured as the number of generations times the size of the offspring population. Besides the presentation of the detailed Tables with the average and standard deviation of the performance achieved by each of the considered AOS methods (*e.g.*, Table 6.15a), the ECDFs are also used to compare the complete performance distribution for each of the winner techniques (*e.g.*, Figure 6.15b).

The stopping criteria are: optimal solution found or maximum number of generations attained. For this latter, a value of 15,000 is used for the OneMax and Long K-Path problems, while 25,000 is used for the Royal Road problem. For the first two problems, the unique solution maintained in the population is initialized to $(0, \ldots, 0)$, while for the Royal Road, the population is uniformly initialized.

In addition to the AP *Operator Selection* technique, combined with all the *Credit Assignment* schemes based on raw values of fitness improvements, the optimal Oracle and the Naive uniform operator selection strategies are used as baseline for both, OneMax and Long K-Path problems. For the Royal Road problem, all of them are used as well, except for the Oracle strategy: as the fitness landscape of this problem includes many paths toward the optimal solution, the Oracle can not be easily accessed. Lastly, the probability of applying each operator was off-line tuned for each problem, by means of the same F-Race procedure (as described in Section 6.2.2). All the possible combinations of probabilities summing up to 1, considering the values ranging in $\{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$, were tried in the off-line tuning phase. The winner configuration of static probabilities is also used as baseline for comparison, being referred to as "Static".

More specific experimental settings, such as the definition of the EA used, as well as

the set of operators automatically controlled by the AOS schemes, will be described in the following, together with the presentation of each problem.

6.4.2 The OneMax Problem

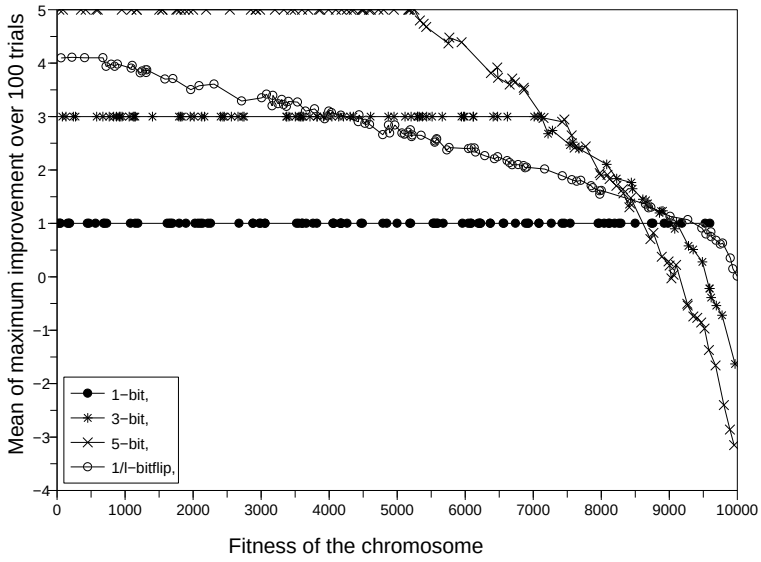
The OneMax problem involves an unimodal fitness function that simply counts the number of "1"s in the binary bit-string that represents the individual solution. The only difficulty comes from the size of the problem; in the presented experiments, the size ℓ of the bitstring is 10,000. Given its simplicity, it is often used to preliminary evaluate new empirical or theoretical methods, being for this reason considered as the "Drosophila of EC".

In order to be assessed on this problem, the AOS schemes are combined with a standard $(1 + \lambda)$-EA (λ offspring are created from the current parent; next parent is the best among the current offspring and parent). Different values for λ were analyzed in [Fialho *et al.*, 2008], achieving similar conclusions; $\lambda = 50$ is used here. The objective is to automatically select between some mutation operators, namely, the standard bit-flip operator (every bit is flipped with probability $1/\ell$), and a set of b-bit mutations (flipping exactly b randomly chosen bits) with $b \in \{1, 3, 5\}$.

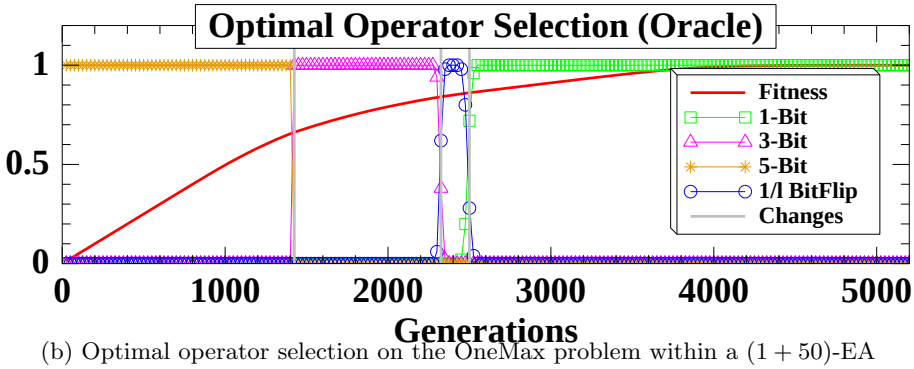
In many respects, the considered setting is still far from being realistic evolutionarily speaking: applying a $(1 + \lambda)$-EA, with $\lambda > 1$ and b-bit mutation operators, is meaningless on the OneMax problem (though it might make more sense on multi-core architectures). It nevertheless confronts the proposed and baseline approaches with the actual difficulties of taming a dynamic system, where the decisions made at a given moment govern the expected benefits of further decisions (the selected operators determine the position of the population and hence the improvement expectation of the operators at further stages of the search), as opposed to the artificial scenarios tackled in Section 6.3.

One main advantage of this kind of "sterile EC-like" environment is to enable the assessment of the AOS approaches by comparison with the performance of the known optimal operator selection. The optimal baseline is provided by the optimal behavior of all operators (computed by means of Monte-Carlo simulations). Figure 6.11a depicts the operator landscape from the perspective of a $(1 + 50)$-EA; for each fitness value of the unique parental individual, we report the fitness gain for the best out of 50 offsprings generated by each of the considered mutation operators, averaged over 100 runs. As can be seen, the trajectory of evolution involves distinct phases with respect to operator dynamics. In *stable* phases, the optimal operator remains the same (though its performance might decrease). For instance, the 5-bit mutation dominates all other operators until around $\mathcal{F}(x) = 6579$, although its performance starts to gradually decrease after $\mathcal{F}(x) = 5300$. In *transition* phases, the established best operator becomes dominated by another one; the 3-bit mutation outperforms the 5-bit after $\mathcal{F}(x) = 6579$, and the 1-bit mutation outperforms the 3-bit after $\mathcal{F}(x) = 8601$. The last phase is a *desert*, where hardly any operator brings any improvement; by being less disruptive, the 1-bit has higher chances of fine-tuning the solution towards the optimum, being thus the preferred operator at this phase.

A clearer view with respect to *Operator Selection* is presented in Figure 6.11b: at each stage of the search, according to the current fitness value of the parent, one of the operators is the optimal operator, and should thus be applied at a rate of 100% (until the



(a) Average fitness gain of operators w.r.t. the fitness of the parent, within a (1+50)-EA applied to the 10,000 bits OneMax problem, averaged over 100 trials.



(b) Optimal operator selection on the OneMax problem within a $(1 + 50)$-EA

Figure 6.11: Different views of the Oracle on the OneMax problem

situation changes). This illustration indeed represents the behavior of the Oracle strategy that was used to achieve the empirical results for the optimal baseline.

Although being a rather simplistic scenario, this operator landscape provided by the OneMax problem enables one to assess the basic skills of an AOS mechanism: the abilities (i) to pick up the best operator and stick to it in stability phases; (ii) to swiftly switch to the next best operator in transition phases; and (iii) to remain efficient during the desert phases. An empirical analyses on this scenario will now be presented.

Empirical Results

The detailed results for the OneMax problem, with the average number of generations (plus the standard deviation) to achieve the optimum and the winner hyper-parameter configuration for each AOS combination, are presented in Table 6.15a. Complementarily,

Table 6.15b depicts the complete distribution of results for each *Operator Selection* technique, with its best *Credit Assignment* scheme and hyper-parameter configuration, in the form of Empirical Cumulative Distribution Functions (ECDFs).

The complete Naive strategy, that uniformly selects between the four available mutation operators, is able to find the optimum in 7955 generations in average. Another common approach is to tune off-line the application rates of each operator: the best Static strategy applies the 5-bit mutation operator at a rate of 20%, and the 1-bit at a rate of 80%, achieving the optimum in roughly 6206 generations. The Oracle strategy (depicted in Figure 6.11b), which represents the complete knowledge about the operator landscape, finds the optimum in 5136 generations in average.

The ECDF plot (Table 6.15b) is bounded on the right by the average performance of the Naive uniform approach. As can be seen in this Figure, 100% of the runs of all the winner AOS combinations achieve the optimum many generations before the Uniform does, all being significantly better than it in average. Besides, with a few exceptions (the combinations involving MAB and SLMAB), all the winner configurations for each *Operator Selection* technique are also able to significantly outperform the baseline method that employs off-line tuned Static probabilities.

Interestingly, several AOS methods are able to achieve a performance statistically equivalent to the Oracle strategy, namely: AP with both Normalized and Absolute versions of the Extreme *Credit Assignment*; and RMAB with any of the rank-based *Credit Assignment* methods based on ranks over the fitness improvements (ΔF), except for the one with NDCG/AUC, which also performs very well but significantly worse, due to the tight standard deviations. In fact, as can be seen in the operator quality landscape depicted in Figure 6.11a, the 5-bit, 3-bit and $1/\ell$ bit-flip mutation operators are rather equivalent, starting from fitness 7000 up to around 9000: hence, these AOS methods are able to achieve optimal performance by controlling the operator applications in very different ways for the fitness values within the mentioned interval. The Normalized Extreme (NormExt)-AP selects different operators in three very well-defined phases (Figure 6.12a), instead of four in the case of the Oracle, efficiently exploiting the 5-bit, then the 3-bit, and finally the 1-bit. The Decay/AUC-RMAB (Figure 6.12e) achieves basically the same performance by exploiting only the 5-bit mutation operator up to 100% at the initial stages of the search, and the 1-bit at the final stages, while for the mentioned fitness interval all the operators are equally explored to some extent. The best configuration for DMAB, implementing the AbsExt *Credit Assignment*, also achieves good performance, although significantly worse than the overall winner. As shown in Figure 6.12b, it also exploits the 5-bit in the initial stages, and explores the 5-bit, 3-bit and bit-flip operators in the middle stages, by means of restarts; however, its performance is degraded by the fact that it is not able to maintain an optimal level of exploitation for the 1-bit operator during the final stages of the search. The MAB with NormExt is also able to efficiently follow the changes; but, in the same way as the DMAB, it gets lost during the final desert phase. A big deception in this scenario is the performance of all the AOS combinations considering the SLMAB *Operator Selection* technique, which shows rather poor performance: a tentative explanation is that it is designed to react very quickly to abrupt changes with respect to the operator qualities (as empirically verified in Section 6.3), while in the OneMax problem the operator qualities

tend to gradually decrease as the search goes on (Figure 6.11a).

It is also worth noticing that the best results are attained by AOS combinations using rather the NormExt or one of the rank-based *Credit Assignment* schemes. This empirical finding clearly confirms that, even in such a simplistic scenario as the OneMax problem, a robust *Credit Assignment* is important in order to achieve good performance: as discussed in Section 5.2.3, it prevents the AOS mechanism from the need of tackling an extra problem, the gradual reduction of the magnitude of the credits, that might greatly affect the performance of the AOS schemes (not to mention its robustness with respect to its hyper-parameters, that will be separately analyzed in Section 6.7.1). However, the comparison-based *Credit Assignment* schemes, *i.e.*, the rank-based methods that assign ranks over the fitness values (F) instead of fitness improvements (Section 5.2.5), which are expected to be the most robust schemes over all, achieve a rather regular performance in this experimental scenario, although still significantly outperforming both Naive and Static baseline approaches. Figure 6.12f depicts the behavior of RMAB with the **F**AUC scheme. As can be seen, surprisingly, there are big variations (in both senses) in the operator selection rates; a tentative interpretation for this very noisy behavior goes as follows. Firstly, the fact that a $(1 + 50)$-EA setting is being used implies that even an improvement of 1 bit will generate a fitness value higher than all the values attained during the previous generation: if this happens in the beginning of the generation, such fitness value will be top-ranked in the *Credit Assignment* sliding window (what does not happen when considering fitness improvements), consequently leading to further trials for the operator used to generate it, being it really the best operator or not. Given the dynamics of the underlying algorithm, this situation, *i.e.*, the exploration of a sub-optimal operator in the first trials of a new generation, could be considered to be rare. The problem in this case lies in the fact that the RMAB, in the way it is conceived (Section 5.3.4), enforces at least one application of each operator every W trials[1]. Hence, all k operators are explored in the initial k steps and once every W steps: as $W = 100$ and the offspring population size $\lambda = 50$, the initial k steps of every 2 generations will always be (coincidentally) exploration trials in this case, consequently leading to the noisy variations presented in the behavior plot roughly every 100 steps.

Finally, this experimental setting was also used to empirically compare the current version of the AUC method with the preliminary one (referred to as AUCv1). As discussed in Section 5.2.4, AUCv1 has normalization issues when considering several operators, and this results in a degraded performance, as presented in the last line of Table 6.15a; while the current version achieves optimal performance, as previously discussed. The behavior plots of RMAB with AUCv1 and FAUCv1 are presented, respectively, in Figures 6.12g and 6.12h: the former erroneously exploits the $1/\ell$ bit-flip during its desert phase, while the latter is totally lost with respect to the *Operator Selection* task.

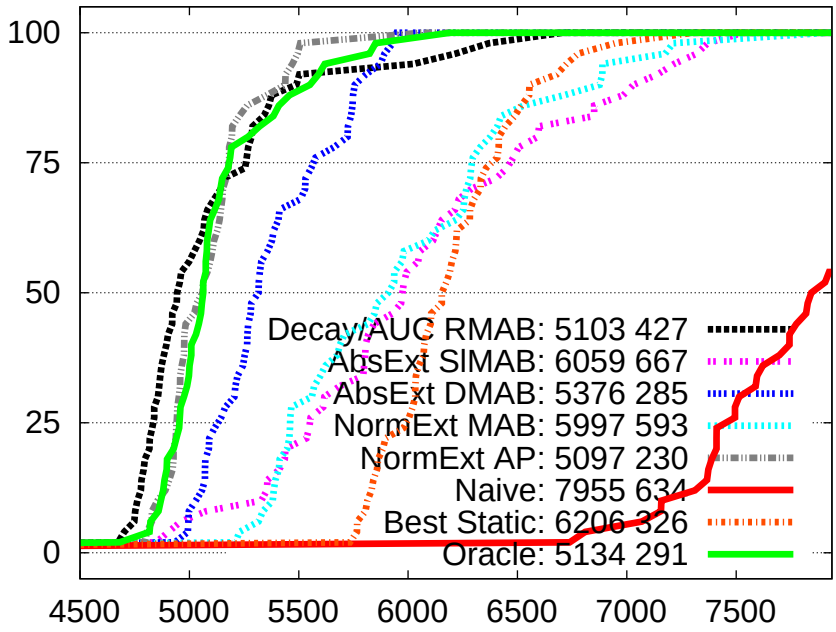
[1]RMAB does not consider n in the MAB formula (Equation 5.11) as the total number of times each operator was applied since the beginning of the search, but rather as the number of times each operator appears in the current sliding window of the *Credit Assignment* scheme. Hence, by using this n, the exploration term of the MAB formula (Equation 5.11) will always ensure that there is at least one application of each operator every W trials.

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	6576 ± 705 ▲ C.01W500	5480 ± 276 ▲ C.5G.1W1	8369 ± 891 C100W1	5718 ± 239 P.05A.6B.1W1
NormIns	6662 ± 961 C.01W500	5444 ± 252 ▲ C.1G.001W1	8013 ± 671 C100W1	5728 ± 204 P.05A.9B.1W1
AbsAvg	8347 ± 596 C.1W500	7494 ± 611 C.5G.001W10	8198 ± 683 C100W50	5750 ± 251 P.05A.3B.3W10
NormAvg	8463 ± 818 C1W100	7193 ± 1614 C.1G.1W10	7903 ± 638 C100W10	5790 ± 226 P.05A.1B.1W10
AbsExt	6059 ± 667 ★ C.5W500	5376 ± 285 ★ C1G100W50	9044 ± 840 C100W50	5123 ± 218 ▲ P0A.9B.1W500
NormExt	6427 ± 597 ▲ C.1W500	5508 ± 823 ▲ C.01G1000W50	5997 ± 593 ★ C1W500	**5097 ± 230 ★** **P0A.9B.6W100**

OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	5103 ± 427 ★ C.01D.9W500	5215 ± 374 ▲ C1D.5W500	5366 ± 478 C.01W500	5231 ± 503 ▲ C.1W100
RMAB (F)	5726 ± 399 C.01D.75W100	5652 ± 644 C.01D.5W500	5796 ± 420 C.1W100	5667 ± 729 C.01W500

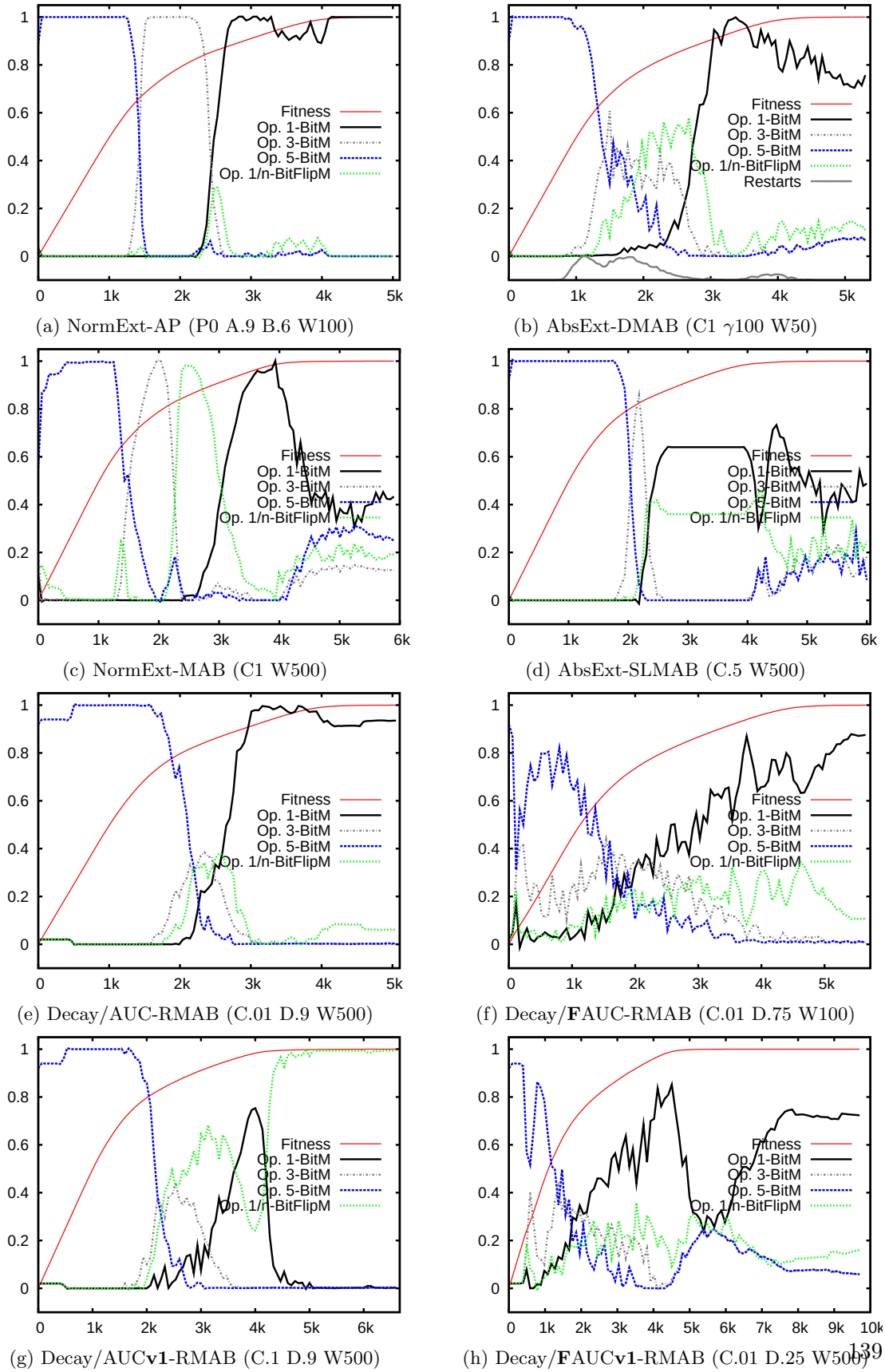
OpSel/Credit	AUCv1 (Decay)	AUCv1 (NDCG)	FAUCv1(Decay)	FAUCv1(NDCG)
RMAB (v1)	6664 ± 631 C.1D.9W500	6741 ± 587 C.01W500	6811 ± 742 C.01D.25W500	6907 ± 708 C.1W500

(a) Average and standard deviation of the number of generations to achieve the optimum



(b) Comparison of Empirical Cumulative Distribution Functions, for each *Operator Selection* technique with its best *Credit Assignment* scheme

Table 6.15: Results on the 10k-bits OneMax problem: objective is to minimize number of generations to achieve the optimum, selecting between 1-bit, 3-bit, 5-bit and $1/\ell$ bit-flip mutation operators within a (1+50)-EA. Baseline performances: Optimal (5134 ± 291), Best Static (1-bit 80% + 5-bit 20% : 6206 ± 326), Naive (7955 ± 634). For the sake of comparison, the performance of the preliminary version of AUC (AUCv1) is also presented.



(a) NormExt-AP (P0 A.9 B.6 W100)

(b) AbsExt-DMAB (C1 γ100 W50)

(c) NormExt-MAB (C1 W500)

(d) AbsExt-SLMAB (C.5 W500)

(e) Decay/AUC-RMAB (C.01 D.9 W500)

(f) Decay/**F**AUC-RMAB (C.01 D.75 W100)

(g) Decay/AUC**v1**-RMAB (C.1 D.9 W500)

(h) Decay/**F**AUC**v1**-RMAB (C.01 D.25 W500)

Figure 6.12: Behavior of AP, DMAB, SLMAB, MAB and RMAB, combined with their best *Credit Assignment* schemes, on the 10k-bits OneMax problem. For the sake of comparison, the behavior of the preliminary version of AUC (AUCv1) is also plotted.

6.4.3 The Long K-Path Problem

Proposed by [Horn *et al.*, 1994], Long Paths are unimodal problems designed to challenge local search algorithms. The optimum can be found by following a path in the fitness landscape, the length of which increases exponentially with respect to the bit-string length ℓ. Accordingly, solving the Long Path using the 1-bit mutation thus requires a computational time that increases exponentially with ℓ; efficient optimization relies on taking shortcuts on this path.

A generalization of Long Path problems was proposed by [Rudolph, 1997], referred to as Long K-Path, where k is the minimal number of bits to be simultaneously flipped in order to take a shortcut on the path. Formally, the Long K-Path can be described as follows [Garnier and Kallel, 2000]:

- The path starts at point $0, \ldots, 0$, with fitness ℓ; the fitness of any point not on the path is the number of its 0 bits;
- Any point on the path has exactly 2 neighbors with Hamming distance 1 on the path; consequently, two consecutive points on the path have a fitness difference of 1;
- Mutating $i < k$ bits of a point on the path leads to a point which is either off the path (hence with a very low fitness), or on the path but only i positions away from the parent point;
- A shortcut is found by mutating the correct k bits (or more), thus with probability at most $p^k(1-p)^{\ell-k}$.
- The length of the path is calculated as $(k+1)2^{(\ell-1)/k} - k + 1$;

Long K-Path problems are defined by recurrence on ℓ. Starting from the problem $P(k, \ell)$, the path associated to problem $P(k, \ell + k)$ is built as the sequence of $(x_i, 0_k)$, where x_i belongs to $P(k, \ell)$ and 0_k is the k-length vector made of 0s; this initial sequence is then linked by a "bridge" to the sequence $(x_{L-i}, 1_k)$, where x_{L-i} ranges in inverse order in $P(k, \ell)$ and 1_k is the k-length vector made of 1s. The bridge is the sequence of (x_L, y_z) where x_L is the last point of path $P(k, \ell)$ and y_z is the k-length vector made of z 0s followed by $k - z$ 1s. To exemplify, the construction of the path $P(3, 10)$ is done as follows. Starting from $P(3, 1) = \{0, 1\}$, the sequence $P(3, 4)$ is created:

$$\underbrace{0000_1; 0001_2}_{S_0}; \underbrace{0011_3; 0111_4}_{Bridge}; \underbrace{1111_5; 1110_6}_{S_1}$$

from which the path $P(3, 7)$ is constructed (Table 6.16a); and finally we arrive to $P(3, 10)$ (Table 6.16b). Going on three more steps with these recursive construction, we arrive to $P(3, 19)$, represented in Figure 6.16c as the number of ones (unitation) versus fitness (red part of the path represents S_0, blue is the $Bridge$, and green is the final sequence S_1).

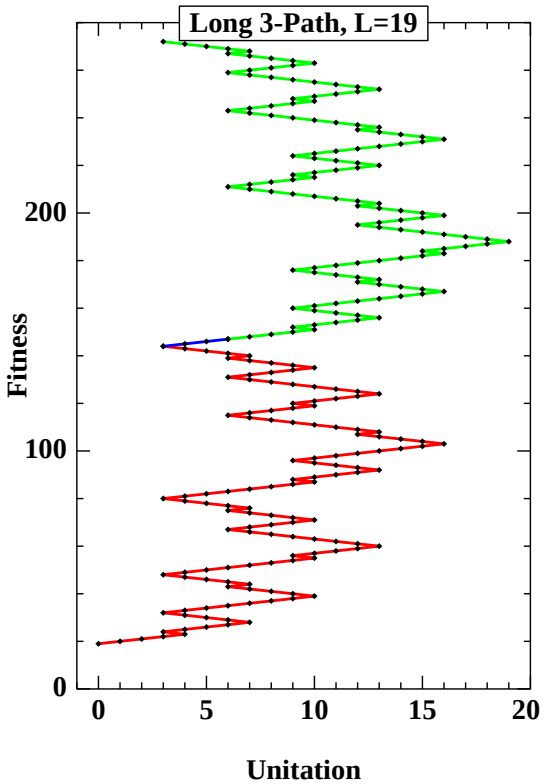
It turns out that the path length decreases as k increases (the original Long Path corresponds to $k = 2$). Nevertheless, the probability of finding a shortcut decreases exponentially with k, and the fastest strategy for $k > \sqrt{\ell}$ is to simply follow the path.

0000000_1	0011110_7	1111110_9
0000001_2	0111110_8	1111111_{10}
0000011_3		1110111_{11}
0000111_4		1110011_{12}
0001111_5		1110001_{13}
0001110_6		1110000_{14}
S_0	*Bridge*	S_1

(a) Path $P(3,7)$

0000000000_1	0011110000_{15}	1111110000_{17}
0000000001_2	0111110000_{16}	1111110001_{18}
0000000011_3		1111110011_{19}
0000000111_4		1111110111_{20}
0000001111_5		1111111111_{21}
0000001110_6		1111111110_{22}
0000011110_7		1110111110_{23}
0000111110_8		1110011110_{24}
0001111110_9		1110001110_{25}
0001111111_{10}		1110001111_{26}
0001110111_{11}		1110000111_{27}
0001110011_{12}		1110000011_{28}
0001110001_{13}		1110000001_{29}
0001110000_{14}		1110000000_{30}
S_0	*Bridge*	S_1

(b) Path $P(3,10)$



(c) Path $P(3,19)$

Table 6.16: Examples of Long 3-Paths of different length.

Otherwise ($k \leq \sqrt{l}$), optimization should provably strive to find the shortcuts; in such cases, *exceptional properties of operators are more relevant to EAs behavior than their average properties* [Garnier and Kallel, 2000].

Along the same lines than the $\mathcal{ART}$ instances analyzed in Section 6.3.3, thus, Long K-Path problems can be seen as yet another case in which one operator constantly gives a small reward (when the parent individual belongs to the path, the 1-bit mutation improves the fitness by 1 with probability $1/\ell$), while all other mutation operators will fail to improve the fitness in most cases, but possibly achieving very high outlier fitness improvements (shortcuts in the path) with a very small probability. This possibility of having outlier fitness improvements was the main motivation for the use of such a scenario in the assessment of AOS schemes by the time we proposed the Extreme *Credit Assignment* [Fialho *et al.*, 2009a; Fialho *et al.*, 2009b] (see Section 5.2.2), which indeed showed to perform better than the Average one for most of the *Operator Selection* techniques considered; these results will be presented in the following, together with the most recently proposed AOS schemes.

The reported experiments consider $k = 3$ with $\ell = 49$. Other problem sizes were also tried, with $\ell \in \{19, 31, 43, 55, 61\}$; they are omitted here, mainly because the conclusions

attained on all of them were roughly the same (except for $\ell = 61$, in which none of the methods was found to be effective, due to the very low probability of finding shortcuts). A $(1+50)$-EA is used here again, with the AOS schemes selecting between some mutation operators. The same operator set used in the OneMax problem (1-bit, 3-bit, 5-bit and $1/\ell$ bit-flip) is considered here, with one additional mutation operator, the k/ℓ bit-flip (flipping each bit with probability $k/\ell = 3/49$ in this case), which is the best operator in this scenario according to theoretical studies [Garnier and Kallel, 2000].

In the same way as for the OneMax problem, the benefit of using this benchmark setting is that it enables the identification of the optimal operator at each point of the path, by means of intensive Monte-Carlo simulations, in order to further compare the AOS approaches with the resulting optimal Oracle strategy. Figure 6.13 shows the average fitness improvement achieved by each of the considered operators, starting from each fitness point on the $P(3, 49)$ path, calculated as for the OneMax problem (best gain out of 50 trials for each operator, averaged over 100 runs).

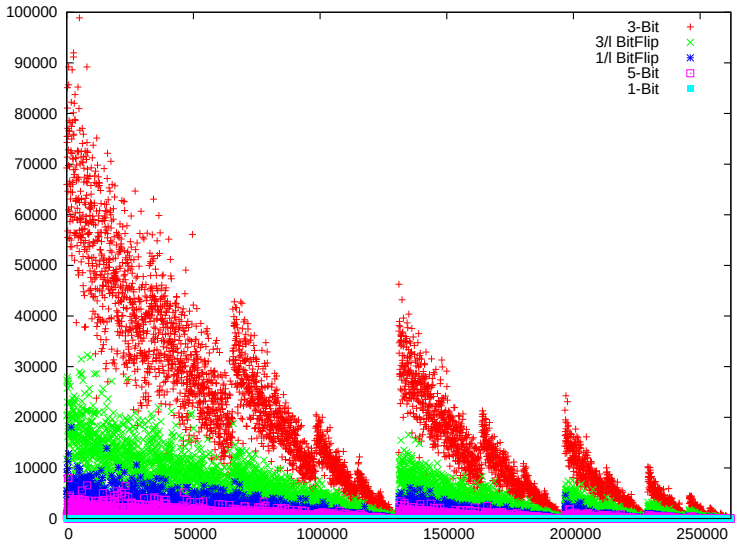


Figure 6.13: Average fitness gain of mutation operators with respect to the fitness of the parent, within a $(1 + 50)$-EA applied to the Long 3-Path problem with $\ell = 49$, averaged over 100 trials. The optimum fitness value in this case is $\mathcal{F} = 262142$.

From this Figure, it can be seen that the 3-bit (or k-bit) operator is the one that receives the highest gains during almost all the path, as it deterministically flips k bits every time it is applied, thus having higher chances of taking a shortcut. For the same reason, the k/n bit-flip comes next, flipping k bits in average. The k-bit operator, however, is able to achieve the optimum just in case it succeeds in taking a shortcut, while the k/n bit-flip can manage to succeed in either cases: this empirically confirms the theoretical findings presented in [Garnier and Kallel, 2000]. It is also important to note that there are two factors controlling the variance of the gains brought by taking shortcuts: the distance to the optimum, and the distance to some transition points found in the path.

The Oracle *Operator Selection* strategy was implemented following the results presented in this Figure: the 3-bit is mostly applied, the 1-bit is used in a few short transition phases and in fine-tuning final phase, while the other operators are also very occasionally applied.

Empirical Results

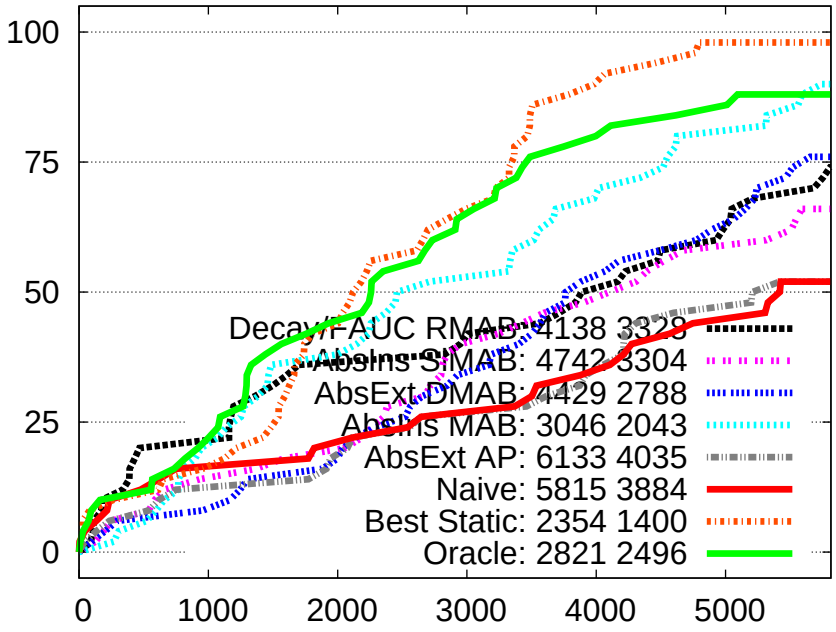
By construction, some runs on the Long K-Path problem can be "lucky" and discover shortcuts in the path, thus yielding large standard deviations in the performance, as shown in the detailed results presented in Table 6.17a. This is true even for the Oracle strategy, which achieves the optimum in 2821 generations in average, but with a standard deviation of 2496. In this case, the ECDFs, shown in Table 6.17b, are much more informative for the analysis of the empirical comparison. As can be seen, for all the considered techniques, there are runs reaching the optimum in the very early steps (close to 0), while many others are not able to achieve the optimum before the average performance of the Naive uniform strategy, which bounds the plot at 5815 generations. With such a big variance, the behavior plots become meaningless: the instant selection rates are averaged over 50 runs, but the good runs attain the optimum very early by taking shortcuts; thus, a behavior plot would be averaging only the longer (hence bad) runs, what does not correspond to the mean behavior of the method.

Interestingly, the off-line tuned approach using Static probabilities, which applies the 1-bit at 20% of the trials and the 3-bit at a rate of 80%, is able to outperform the Oracle strategy. The Oracle explores mostly the same operators, but in a fixed manner: the 3-bit is used for some fitness ranges due to its high probability of finding a shortcut, while the 1-bit is used only in transition phases where no shortcut is possible. In practice, however, it seems that using the 1-bit at a fixed small rate is more beneficial: although providing very small improvements (1 by 1 in fact), its probability to improve the fitness is high ($1/\ell$) when compared to the probability of taking outlier shortcuts in the path.

The winner AOS combination in this case is the MAB, which, in the ECDF plot (Table 6.17b) is the only method able to follow the performance of both Oracle and Static baseline methods. As previously discussed, the MAB (as well as the other bandit-based approaches) does some averaging on the update of the empirical quality estimates (Equation 5.12) for each operator. Thus, even when using the AbsIns *Credit Assignment*, it takes into account some history of the operator performance, consequently not forgetting it very quickly in such a noisy environment. Additionally, it is known that MAB is the slowest *Operator Selection* technique with respect to adaptation between the bandit-based methods considered here (as verified in Section 6.3); this seems to be a beneficial characteristic in this case: the 3-bit should continue to be exploited as much as possible even if some other operator appears to be very good from time to time, because it has a much higher probability of taking shortcuts in a Long Path with $k = 3$, as previously pointed out. The RMAB with FAUC (fitness values, comparison-based) follows the same trend in around 40% of the runs, but its global picture is rather similar to both AbsExt-DMAB and AbsIns-SLMAB. Lastly, the AP is not able to cope well with Long K-Paths: its best results are obtained with $p_{min} = 0.2$, what is equivalent to the Naive uniform selection of operators in this case, as 5 operators are considered.

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	4742 ± 3304 ★ C.5W500	4680 ± 3714 ▲ C100G100W1	**3046 ± 2043 ★** **C100W1**	6133 ± 4035 ▲ P.2A.1B.1W1
NormIns	5601 ± 3161 ▲ C100W500	5355 ± 3880 ▲ C100G.001W1	5429 ± 3912 ▲ C100W1	6133 ± 4035 ▲ P.2A.1B.1W1
AbsAvg	4985 ± 2942 ▲ C1W10	5509 ± 3180 ▲ C1G.1W10	4564 ± 3004 ▲ C10W10	8303 ± 5242 ▲ P.1A.1B.1W50
NormAvg	5337 ± 3120 ▲ C5W500	4829 ± 3218 ▲ C.01G1W10	4851 ± 3219 ▲ C.01W10	7430 ± 5152 ▲ P.1A.1B.1W50
AbsExt	4828 ± 3520 ▲ C100W500	4429 ± 2788 ★ C100G100W100	4596 ± 3034 ▲ C100W10	6133 ± 4035 ★ P.2A.1B.1W100
NormExt	5442 ± 3662 ▲ C100W500	4505 ± 3349 ▲ C10G.1W50	5529 ± 3273 C100W10	6133 ± 4035 ▲ P.2A.1B.1W100
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	5005 ± 3723 ▲ C5D.75W100	4931 ± 3307 ▲ C10D.5W500	5018 ± 3413 ▲ C100W50	5032 ± 3751 ▲ C100W500
RMAB (F)	4138 ± 3328 ★ C10D.9W50	4139 ± 3328 ▲ C100D.75W50	4723 ± 3820 ▲ C100W50	5071 ± 4152 ▲ C100W50

(a) Average and standard deviation of the number of generations to achieve the optimum



(b) Comparison of Empirical Cumulative Distribution Functions, for each *Operator Selection* technique with its best *Credit Assignment* scheme

Table 6.17: Results on the Long 3-Path ($\ell = 49$) problem: objective is to minimize number of generations to achieve the optimum, selecting between 1-bit, 3-bit, 5-bit, $1/\ell$ bit-flip and $3/\ell$ bit-flip mutation operators within a (1+50)-EA. Baseline performances: Oracle (2821 ± 2496), Naive (5815 ± 3884), Best Static (1-bit 20% + 3-bit 80% : 2354 ± 1400)

6.4.4 The Royal Road Problem

The Royal Road (RR) is an optimization problem that was intentionally created to be easy for GAs [Mitchell *et al.*, 1992] (with the crossover operators exploring the "building blocks" of the function), while being difficult for hill-climbing algorithms. Due to unexpected difficulties (the so-called hitch-hiking phenomenon), a revised version was later proposed in [Holland, 1993] and analyzed in [Jones, 1994]. The revised version is the one considered in this work.

The solutions are represented as bit-strings. Each bit-string is composed of 2^k regions, referred to as lower-level (or level 0) *schemata*. Higher level schemata are formed by combining pairs of lower-level ones, as shown in Table **??**.

Level 0:	$\{B_0\}, \{B_1\}, \{B_2\}, \{B_3\}, \{B_4\}, \{B_5\}, \{B_6\}, \ldots, \{B_{12}\}, \{B_{13}\}, \{B_{14}\}, \{B_{15}\}$
Level 1:	$\{B_0, B_1\}, \{B_2, B_3\}, \{B_4, B_5\}, \ldots, \{B_{10}, B_{11}\}, \{B_{12}, B_{13}\}, \{B_{14}, B_{15}\}$
Level 2:	$\{B_0, B_1, B_2, B_3\}, \{B_4, B_5, B_6, B_7\}, \ldots, \{B_{12}, B_{13}, B_{14}, B_{15}\}$
Level 3:	$\{B_0, B_1, \ldots, B_7\}, \{B_8, B_9, \ldots, B_{15}\}$
Level 4:	$\{B_0, B_1, \ldots, B_{15}\}$

Table 6.18: Example of constructions of higher order schemata from lower order ones on the Royal Road problem

Formally, a higher level L has 2^{k-L} schemata composed by 2^L first-level ones (the building-blocks, supposedly defining a crossover-friendly landscape). Each first-level schema is further divided into a *block* and a *gap* string, of respective lengths b and g. A bit-string is thus represented by $2^k \times (b + g)$ bits.

For the calculation of the fitness of a candidate solution (bit-string), each first-level schema is independently evaluated, with the fitness resulting in the sum of the evaluations of all the schemata. Only the block region of each low level schema is considered, the gap region is completely ignored. The fitness is measured by the PART function or by the BONUS function, as follows. The PART function computes the number z of correct bits in the b-length block, resulting in a function value of $(z \times v)$ if $(z < m)$ and $((b - z) \times v)$ for $(m < z < b)$, where m is a threshold that tunes the level of local deception in the function (see Figure 6.14). The completed blocks in the bit-string, *i.e.*, the ones that have $z = b$, are evaluated by the BONUS function instead, which accounts a score of u^* for the first block to be completed, and u for the additional ones.

The Royal Road was found to be another interesting scenario to empirically analyze the AOS combinations within a real evolutionary algorithm as, by considering common crossover operators, the following can be stated (intuitively, and confirmed in [Quick *et al.*, 1996]): the uniform crossover is the best operator during the initial evolution stages (exploration), 1-point crossover is the best in the final stages (exploitation), while the 4-point crossover is the best in-between. This operator set was used for the experiments that will be presented in the following, with the addition of two other operators, the 2-point crossover, and a disruptive bit-flip mutation operator that flips 8 bits on average (and hence possibly one block); after every crossover application, a mutation operator was also systematically applied, flipping each bit with a probability of 1%. These

operators were applied within a (100,100)-GA with weak elitism, *i.e.*, at every generation, the entire population of 100 individuals is completely replaced by the newly generated 100 offspring, with the possible exception of the best parent, which is maintained (and the worst offspring is removed) if better than the best offspring. The parental selection mechanism used was the tournament (Section 2.3.3), with size 2.

The problem function was defined using the default parameter values proposed by Holland [Holland, 1993]: $k = 4$, $b = 8$, $g = 7$, $m = 4$, $v = 0.02$, $u^* = 1.0$ and $u = 0.3$. The parameter $m = 4$ defines a medium level of deception; the fully deceptive case ($m = 1$) and the not deceptive one ($m = 7$) were also investigated, but the former was found too difficult to be solved within the given budget of 25,000 generations, while the latter was too easy, thus not enabling any distinction to be made between the AOS schemes. Figure 6.14 illustrates a comparison of these 3 different levels of deceptivity for the first 30 bits of this problem setting, on a unitation (number of 1s) versus fitness plot. With 2^k regions involving $(b + g)$ bits, the total dimension of the considered search space accounts to 240 bits.

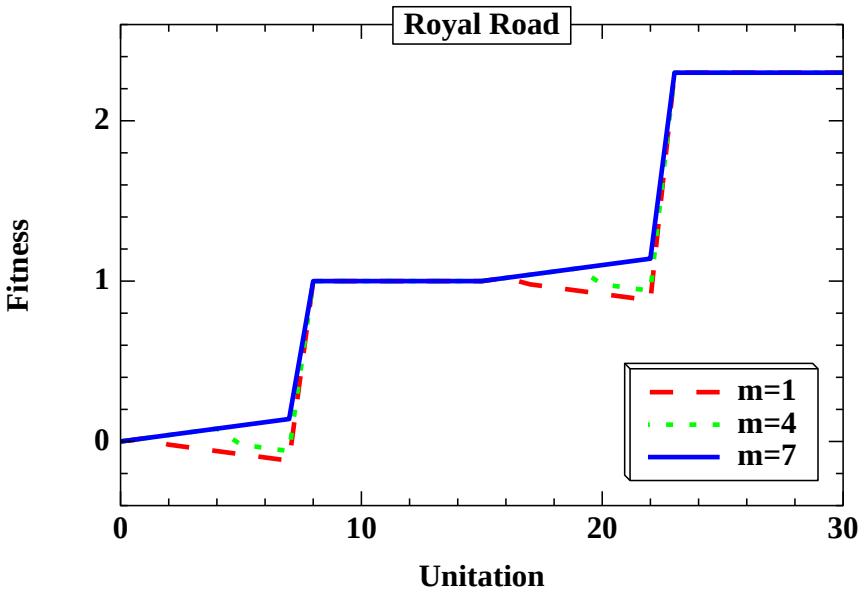


Figure 6.14: Different levels of deceptivity on the Royal Road problem, varying m and using the default values for the other parameters.

Empirical Results

The behavior of each operator is very difficult to guess on the Royal Road problem. Despite the 1/30 mutation operator, the other 4 crossover operators (specially the x-points ones) tend to present a similar behavior, all exploring the building blocks of the intentionally designed search space. Besides, there is no "fine-tuning operator" between them: even the 1-point crossover substantially modifies the solution to which it is applied to, making it easier to miss the target, thus explaining the high variance of the detailed performance

results shown in Table 6.19a. As can be seen, almost all AOS combinations are statistically equivalent to the best. For this reason, as for the Long K-Path problem, ECDFs (Table 6.19b) are used in order to have a more complete view of the performance distribution for each *Operator Selection* technique with its better *Credit Assignment* scheme.

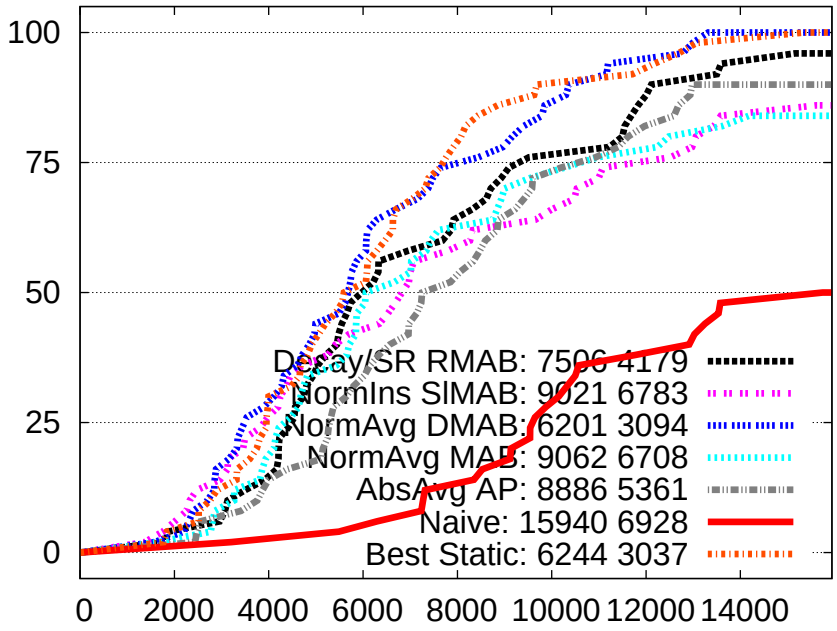
Despite the big variance, all methods achieve the optimum faster than the average performance of the Naive uniform selection strategy in at least 80% of the cases. Notably, the best Static strategy found for this problem is the use of a single operator at a rate of 100%, the 4-point crossover, which achieves the optimum in 6244 generations in average. Several other configurations using different combinations of the 1-point, 2-point and 4-point operators are also able to achieve equivalent performance; for instance, 1-point at 20%, 2-point at 20% and 4-point at 60% achieves the optimum in 6679 ± 4278 generations. Hence, in order to achieve reasonable performance on this experimental setting, an AOS method should "simply" be capable of discarding the 1/30 mutation and the uniform crossover operators; the way the other three operators are used does not matter much. To confirm this assumption, additional experiments were done for the Naive uniform strategy considering only these 3 operators: the optimum is found in 7066 ± 4215 generations, a performance better than (although still equivalent to) those obtained by most of the AOS methods.

The only AOS combination able to follow closely the performance of the Static baseline up to 100% of the trials is the DMAB with, surprisingly, the Normalized Average (NormAvg) *Credit Assignment*. It is worth noticing that here, again, the Normalized outperform the Absolute for the different kinds of *Credit Assignment* in most cases, with the rank-based schemes also presenting reasonable performance. This is also the first case in which the RMAB with the rank-based SR outperforms (but not significantly) its combination with the AUC *Credit Assignment*, with both versions based on fitness improvements and on fitness values (FSR) achieving almost the same performance.

Concerning specifically the *Operator Selection* techniques, the DMAB and RMAB, with their corresponding best *Credit Assignment* schemes, greatly outperform in terms of average performance (but not significantly due to the mentioned high variance) all the combinations involving MAB, SLMAB and AP. It is important to note that for most of the winner configurations of the bandit-based approaches, a very small value is used for the scaling factor C; accordingly, very small values are used for the adaptation and learning rates of AP. This indicates that these techniques are rarely adapting to exploit other operators, mostly exploiting the first operator that they found to be the best. As previously discussed, if this operator is one of the x-point crossover operators, this choice will not greatly affect their performances.

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsIns	10385 ± 4989 ▲ C5W500	8762 ± 5750 ▲ C.01G.01W1	11211 ± 8198 ▲ C.01W1	10681 ± 7048 ▲ P.05A.9B.1W1
NormIns	9021 ± 6783 ★ C.01W50	8538 ± 5388 ▲ C.1G1W1	9548 ± 6485 ▲ C.1W1	10492 ± 6327 ▲ P0A.3B.3W1
AbsAvg	10612 ± 5266 ▲ C1W10	12220 ± 7113 C.5G1000W10	11511 ± 8233 ▲ C.01W10	8886 ± 5361 ★ P0A.1B.1W10
NormAvg	11241 ± 7182 ▲ C.01W10	**6201 ± 3094 ★ C.01G.001W10**	9062 ± 6708 ★ C.01W10	9117 ± 5490 ▲ P.05A.1B.9W10
AbsExt	9790 ± 6019 ▲ C.01W50	10120 ± 6781 ▲ C.01G.01W50	10219 ± 7866 ▲ C.1W50	10860 ± 6428 ▲ P.05A.1B.9W50
NormExt	9780 ± 6359 ▲ C.1W50	8699 ± 5260 ▲ C10G1000W10	9830 ± 5557 ▲ C1W100	9709 ± 7079 ▲ P0A.1B.9W50
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	8749 ± 4640 ▲ C.01D1W100	7506 ± 4179 ★ C1D1W500	9568 ± 5367 ▲ C.1W100	10346 ± 6200 ▲ C.1W50
RMAB (F)	8206 ± 4057 ▲ C.01D.5W100	7564 ± 4282 ▲ C1D1W500	8129 ± 4453 ▲ C.1W100	8508 ± 5595 ▲ C.5W500

(a) Average and standard deviation of the number of generations to achieve the optimum



(b) Comparison of Empirical Cumulative Distribution Functions, for each *Operator Selection* technique with its best *Credit Assignment* scheme

Table 6.19: Results on the Royal Road ($m = 4$) problem: objective is to minimize number of generations to achieve the optimum, selecting between 1-point, 2-point, 4-point and uniform crossover operators, and 1/30 bit-flip mutation operator, within a (100,100)-EA with weak elitism. Baseline performances: Oracle not available, Naive (15940 ± 6928), Best Static (4-point 100% : 6244 ± 3037).

6.4.5 Discussion

The benchmark optimization problems considered in this Section enabled a preliminary analysis of the AOS methods in practice, selecting between real evolutionary operators, based on the feedback given by real (although still artificially created) fitness landscapes and by the trajectory taken by the EA in the search space. The OneMax is a very simple problem, but its use in these experimental setting was of great value, as it provided a very detailed and complete behavioral analysis of each AOS method. The empirical analyses on the Long K-Path and Royal Road problems did not provide the same level of information, but they challenged the AOS methods in different situations that might happen in real cases. In the Long K-Path case, only one operator should be mostly exploited at any time, in order to possibly take very rewarding shortcuts in the path; while in the Royal Road two out of five operators should be discarded, the other three being equally beneficial.

For each of these problems, the AOS methods were compared to three non-adaptive baseline approaches, namely, the Naive, the Oracle, and the Static strategies, defining three different levels of available knowledge. The Naive strategy, as its name says, represents the situation in which nothing is known about the performance of the operators on the problem at hand; the Oracle represents the complete detailed information about their performances with respect to each fitness value. While the former strategy is a rather straightforward choice when there is no time for a deeper analysis, the latter strategy can be precisely assessed only in simple problems such as the OneMax one; hence, it is not a valid choice in the real world. In the middle of these two approaches, there is the Static strategy, which is the approach most commonly used since the very early days of applied research in the area; it requires a reasonable level of knowledge that can be gathered whenever a few runs are affordable, in order to find the best off-line tuned static approach.

Although requiring a preliminary off-line tuning of their hyper-parameters, compared to these baseline approaches, the best AOS methods were able to achieve equivalent performance to the Oracle behavior on both OneMax and Long K-Path problems. They also showed to be able to significantly outperform the Naive approach on all the problems, although the high variance of the results found for the Long K-Path and the Royal Road problems. And concerning the Static strategy, it was also significantly outperformed on the OneMax scenario; but in the Long K-Path, it surprisingly outperformed the Oracle strategy, significantly outperforming all the AOS methods by using only two out of the four available operators; while for the Royal Road, the best Static strategy found uses only one out of five operators, with some AOS methods showing equivalent performance.

In what concerns the *Operator Selection* techniques, the DMAB and RMAB showed to efficiently and consistently improve over the standard MAB approach, except for the outlier Long K-Path scenario, in which the standard MAB is surprisingly the winner: its slower adaptation seems to be beneficial in such a noisy scenario. The SLMAB, however, was not able to outperform the standard MAB in any of the cases, while in the $\mathcal{ART}$ scenarios it was always between the overall winners; a tentative of explanation for this deception is that its update mechanism is designed to adapt quickly to very abrupt changes in the operator qualities, what is not the case in this experimental setting. The last *Operator Selection* technique, the baseline AP, is top-ranked only on the OneMax scenario; on the

Long K-Path its best performance is equivalent to the Naive approach, and on the Royal Road it is also outperformed by all bandit-based approaches.

Finally, in these problems, differently from the $\mathcal{ART}$ scenarios, the benefits brought by the use of more robust *Credit Assignment* schemes could be highlighted to some extent. In the OneMax and Royal Road problems, the Normalized versions of the *Credit Assignment* schemes based on the raw values of fitness improvements outperformed the Absolute versions in most cases; in the Long K-Path, the situation is the opposite, as in this scenario the magnitude of the outlier improvements achieved are very important. In either cases, most of the several available options for the rank-based *Credit Assignment* schemes were top-ranked, performing statistically equivalent to the winner configurations. Still, as for the $\mathcal{ART}$ problems, such analysis compared the performance of each AOS combination with its best hyper-parameter setting found by a preliminary off-line tuning procedure for each problem. Thus, this gain in performance provided by the rank-based and normalized schemes were attained only by the minimization of one of the issues that motivated their proposal, that of providing rewards at the same value range during all the search process, as discussed in Section 5.2.3. The second and consequent benefit, that of providing a robust behavior with respect to the hyper-parameters of the AOS method, will be separately analyzed in Section 6.7.

6.5 Collaboration On Satisfiability Problems

The preliminary complete AOS combination proposed in our work involves the Absolute Extreme (AbsExt) *Credit Assignment* scheme (Section 5.2.2) with the Dynamic Multi-Armed Bandit (DMAB) *Operator Selection* technique (Section 5.3.2), which will be simply referred to as Ex-DMAB in this Section, for the sake of brevity. While working on its further assessment on different benchmark scenarios, we established a collaboration with Université d'Angers, France, published in [Maturana *et al.*, 2009a; Maturana *et al.*, 2010a], whose results will be surveyed in this Section.

The combination of Ex-DMAB with the Compass *Credit Assignment* (Section 4.3.4) will be described in Section 6.5.1. This AOS combination was assessed in the light of Boolean Satisfiability (SAT) problems, which will be presented in Section 6.5.2. Sections 6.5.3 and 6.5.4 will describe, respectively, the specific experimental settings and the off-line tuning procedure used in this work. Finally, the empirical results will be presented in Section 6.5.5, with a concluding discussion in Section 6.5.6.

6.5.1 Compass + Ex-DMAB = ExCoDyMAB

The Compass-based AOS technique, proposed in [Maturana and Saubion, 2008a], is in fact the combination of an engineered *Credit Assignment* mechanism referred to as Compass, which measures the effect of operators application taking into account fitness, diversity and CPU time (as described in Section 4.3.4), with a rather simple *Operator Selection* mechanism, the Probability Matching (PM), presented in Section 4.4.1. At the same time, the Ex-DMAB AOS technique is the combination of a simple *Credit Assignment* scheme, the Extreme value out of the recent fitness improvements achieved by the operator (see

Section 5.2.2), with an efficient *Operator Selection* mechanism, the DMAB, presented in Section 5.3.2.

From this brief review, it becomes clear that both AOS approaches have complementary strengths and weaknesses: Compass might enable DMAB to be efficiently applied to multi-modal problems, while Ex-DMAB might provide to Compass a more efficient *Operator Selection* mechanism, while also improving it by the use of the Extreme paradigm. However, even though merging both modules can be done in straightforward manner, some important issues need to be further explored:

- Compass uses sliding windows in the "impact evaluation" stage (see Figure 4.1), outputting a unique value; while Ex-DMAB keeps a sliding window in the *Credit Assignment* stage, from which it extracts the maximum or Extreme values. Should we keep both windows, or would it degrade or disappear with the interesting characteristics provided by Compass? And if only one of these windows is kept, which one should it be? From here on, these two windows will be respectively referred to as $W1$ and $W2$.

- Another issue concerning the sliding windows is that of what should be their output. Originally, the output of Compass $W1$ is the Average over the impacts measured after the most recent applications [Maturana and Saubion, 2008a]; a simpler approach would be the Instantaneous value, *i.e.*, no window at all. Ex-DMAB uses the Extreme *Credit Assignment*, which was successfully validated in the scope of different artificial (Section 6.3) and unimodal (Section 6.4) benchmark problems. But would these results also hold in such a completely different setting?

- The last issue concerns the other hyper-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 to propose some robust default values.

The resulting combination of Ex-DMAB and Compass is referred to as Extreme Compass - DMAB (ExCoDyMAB). An empirical analysis of the discussed issues will be presented in the following.

6.5.2 SAT Problems

The ExCoDyMAB AOS method has been assessed within an EA applied to the well-known combinatorial Boolean Satisfiability (SAT) problem [Cook, 1971], which consists in assigning values to binary variables in order to satisfy a Boolean formula.

Formally, an instance of the SAT problem is defined by a set of Boolean variables $\mathcal{X} = \{x_1, \ldots, x_n\}$ and a Boolean formula $\mathcal{F}\colon \{0,1\}^n \rightarrow \{0,1\}$. The formula is said to be satisfiable if there exists an assignment $v\colon \mathcal{X} \rightarrow \{0,1\}^n$ 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, and also due to its very general boolean (bit-string) representation, 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 very diverse set of fitness landscapes with different characteristics.

Table 6.20 shows the instances used here, extracted from the SATLIB [Hoos and Stützle, 2000] and from the SAT-Race 2006 [Sinz *et al.*, 2006], pointing out whether they are satisfiable or not, their family, and the number of variables and clauses they involve.

	Problem	Sat?	# Vars.	# Clauses	Family
SATLIB	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
SAT-Race'06	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

Table 6.20: SAT instances used in the empirical assessment of ExCoDyMAB

6.5.3 Experimental Settings

The ExCoDyMAB is applied to an EA, that uses a standard binary representation (one bit per boolean variable) to represent each solution. As in [Maturana and Saubion, 2008a], 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, further exploiting the best operator. The very heterogeneous operator set is constituted by the following operators:

- *1-point Crossover* randomly chooses two individuals and a random position, and exchanges their first and second parts.
- *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 important to note that this is a local search operator, which 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, while improvements can be found.

The parental selection mechanism is the steady-state, defined in Section 2.3.3: after the generation of each offspring, the worst individual in the population is immediately replaced (except when the 1-point Crossover operator is applied, in which case the best offspring replaces the worst parent). The population size (3) and maximum number of generations (5000 – the only stopping criterion) were arbitrarily fixed.

6.5.4 Architecture definition and tuning of hyper-parameters

In order to efficiently integrate Compass and Ex-DMAB, some open issues were discussed in Section 6.5.1. The definition of these components, as well as the off-line tuning of the other hyper-parameters, will be now analyzed in turn.

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 were tried for each window (note that the Normalized versions were not considered in this work):

- Instantaneous (I) value, i.e., no sliding window.
- Average (A) value from stored measures;
- Extreme (E) value (maximum) from stored values (except for the execution time, kept in $W1$, as the extreme of this measure would not make sense);

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

- The size of Compass window of impact measures $W1$, and the size of the sliding window of the outputs of the *Credit Assignment* scheme $W2$.
- The Compass angle Θ, that defines the tradeoff between the EA exploration and exploitation.
- The DMAB scaling C parameter, that defines the tradeoff between exploration and exploitation at the operator-selection level.
- The DMAB γ parameter, the threshold of the change detection test that triggers the restarts.

The range of values tried for the different hyper-parameters are defined 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 $\pi/4$, as preliminary experiments have shown that a different value causes a positive-feedback phenomenon[2], that moves the EA to an extreme behavior. For instance, Figure 6.15 shows the curve of the best fitness found, with respect to the number of time steps elapsed, when using different values of Θ for the original Compass AOS combination applied to the *par16-1* instance, averaged over 50 runs. Note that all values below 0.25π tend to produce a similar erratic behavior, while values above 0.25 have a poor improvement rate.

The same F-Race off-line tuning procedure, described in Section 6.2.2, was used for the tuning of these hyper-parameters. 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}. All 22 SAT instances listed in Table 6.20 have been considered for the final empirical comparison, but only 7 of them were taken into account for this off-line tuning phase. This sub-set, marked with an asterisk in Table 6.20, was chosen among the hardest instances with short enough running times, reducing the experimental cost for the platform definition. Tuning the hyper-parameters on a small set of instances and testing them further on "unseen" instances witnesses the generality of the tuned parameters.

6.5.5 Empirical Results

At the end of the Racing, 4 configurations were still active in the process, which are presented in Table 6.21. These results clearly indicate that the most important sliding window is $W1$, *i.e.*, the Compass window for the impact measures, 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 impact measures assessed), 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 by the proposal of the Extreme *Credit Assignment*, described in Section 5.2.2. Besides, the size of 10 for $W1$ could be interpreted by the following reasoning.

[2]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.

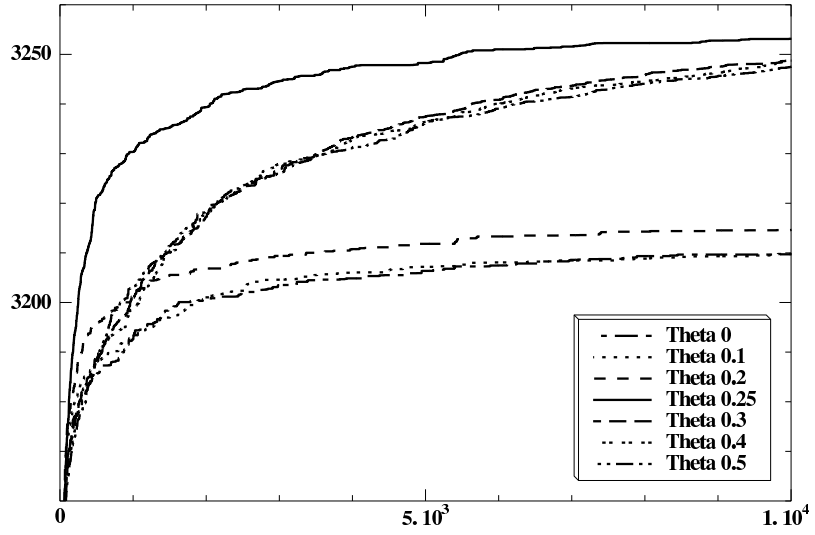


Figure 6.15: Curve of the best fitness found in relation to the number of time steps elapsed, for different values of Θ on Compass applied to the *par16-1* instance, averaged over 50 runs.

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

Table 6.21: Racing survivors for ExCoDyMAB hyper-parameters tuning

With the Extreme policy, a larger $W1$ 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, up to $W1 = 1$ (i.e., the same as choosing the Instantaneous policy) would quickly forget these "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 this hyper-parameter.

To check the generality of those parameters, 50 runs were performed on the 22 SAT instances with each of the 4 configurations, promoting an empirical comparison between them, and also verifying their performances in relation to the baseline methods: the original combinations of Compass and Ex-DMAB (including a Racing phase for Ex-DMAB similar to that of ExCoDyMAB), and the Naive uniform selection of operators. The results of this comparison are summarized in Table 6.22. 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 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.

	Compass	Ex-DMAB	Naive	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
Naive	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

Table 6.22: 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.

After this analysis, between all the four survivors of the Racing procedure, the configuration "C" was found to be the best for ExCoDyMAB, and was thus used for further empirical comparison with the baseline techniques, namely, the original Compass-PM and Ex-DMAB AOS combinations, and the Naive uniform choice. The results are presented in Table 6.23. The columns show the mean number of false clauses after 5000 function evaluations, averaged over 50 runs, and the standard deviation between parentheses. The best results for each instance are highlighted in **bold-face**. As can be seen, ExCoDyMAB outperforms the other techniques in the vast majority of the cases. These results will be further discussed in the following.

6.5.6 Discussion

The dominance of ExCoDyMAB is overwhelming, and confirms the hypothesis that motivated the combination of both Compass and DMAB approaches. These latter approaches alone, within their respective original combinations, present a performance roughly equivalent between each other on this experimental setting, and clearly inferior to the newly combined one, ExCoDyMAB – though still outperforming in turn the Naive uniform selection policy.

Another interesting point is the seemingly good generalization capacity of ExCoDyMAB with respect to its hyper-parameters: the best configurations found by F-Race on the 7 "training" instances showed to perform also very well when solving the other 15 unseen instances. Moreover, the credits assigned by Compass are normalized by construction, and this might result into a more robust technique with respect to the configuration of its hyper-parameters. But this deserves further analysis; in the meantime, the main drawback of this combination is still that of needing to count with a preliminary expensive off-line tuning phase in order to achieve reasonable performance, specially in what concerns the DMAB hyper-parameters.

	Method / Problem	ExCoDyMAB (C)	Compass	Ex-DMAB	Naive
SATLIB	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)
SAT-Race'06	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)

Table 6.23: Comparative results on the 22 SAT instances: average (std dev.) number of false clauses (over 50 runs)

Note that this work was done before the proposal by us of the more robust rank-based AOS approaches. In the same way, the Compass authors have come up with more efficient *Credit Assignment* schemes that also integrate both impact measures, based on the *Pareto Front* paradigm [Maturana *et al.*, 2010b]. As a further work, a combination of these newly proposed components will be analyzed on this scenario, hopefully achieving better results while showing to be more robust with respect to its hyper-parameters (or cheaper in relation to their off-line tuning).

It is also important to remember, as previously mentioned, that the purpose of this work was not to build an overwhelming SAT solver, but rather to experiment and validate the ExCoDyMAB as an AOS technique with an EA solving a general difficult and highly multi-modal combinatorial problem. The main interesting result is that this set of benchmarks was difficult enough to highlight the benefits of using the proposed combination of Compass and Ex-DMAB rather than either separately – or than the 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.

Finally, although the results presented in Table 6.23 show that a basic EA using rather naive operators can indeed solve some instances, competing for SAT Race implies using highly specialized operators, and possibly problem-dependent knowledge, as done in [Wei *et al.*, 2008]. We are currently working on this in collaboration with University of British Columbia, the AOS schemes choosing between state-of-the-art heuristics for variable selection; however, by the time this manuscript is being written, there are no conclusive results yet.

6.6 On Continuous Benchmark Problems

The experimental results surveyed in this Chapter up to now considered the AOS schemes coupled with a Genetic Algorithm, and applied to artificial, boolean benchmark, and SAT problems. In order to analyze the applicability of such methods in a totally different context, we will present in this Section an empirical analysis of AOS schemes selecting between some mutation strategies within a different EA, the Differential Evolution (DE) algorithm (described in Section 2.4.5), applied to continuous optimization problems. In this context, AOS is also sometimes referred to as Adaptive Strategy Selection (AdapSS) [Gong *et al.*, 2010a; Fialho *et al.*, 2010b].

The experimental framework used in these experiments will be introduced in Section 6.6.1. The specific experimental settings will be presented in Section 6.6.2, while Section 6.6.4 will survey the empirical results. Finally, Section 6.6.5 will discuss the findings and point out possible directions for further work. The results that will be analyzed here were partially published in [Fialho *et al.*, 2010b; Fialho and Ros, 2010].

6.6.1 Black-Box Optimization Benchmarking

The Black-Box Optimization Benchmarking (BBOB) is a workshop held yearly during the ACM Genetic and Evolutionary Computation Conference (GECCO), starting from 2009 [Hansen *et al.*, 2010b]. Partly organized by some members of the Project-team TAO, INRIA Saclay - Île-de-France, the main objective of this workshop is to present and discuss empirical comparisons of different optimization algorithms in the continuous domain, using a common experimental framework.

As a result of this initiative, important contributions have been made to the research field of empirical analysis of continuous optimizers. Firstly, the BBOB framework provides: two well-defined and documented sets of benchmark functions, a noiseless and a noisy one; an experimental set-up [Hansen *et al.*, 2010a] for analyzing the algorithms in several dimensions and function classes; and some post-processing scripts to generate graphs and tables to assist the user into the analysis of the performance data. Thus, in case one wants to empirically assess a given optimization algorithm, this task is greatly facilitated by the use of the BBOB framework: the user only needs to interface his optimization algorithm with the framework, allocate some CPU-time, launch some runs, and finally do the post-processing with the aid of the available scripts. Accordingly, this experimental framework can be (and should be) seen as a standard for the empirical analysis of optimization algorithms: by using it, newly proposed algorithms, or new improvements to existing

algorithms, can be easily compared to state-of-the-art methods in a rigorous scientific manner.

The empirical analysis that will be presented in this Section has greatly benefited from the use of this experimental framework. More details will be given in the following.

6.6.2 Experimental Settings

The goal of the experiments that will be presented here is to assess the comparative performances of the AOS schemes when coupled with the standard version of the Differential Evolution algorithm [Storn and Price, 1997], the only difference regarding the way the mutation strategies are selected. As described in Section 2.4.5, the DE algorithm is governed by three parameters: NP, F and CR, respectively denoting the population size, the mutation scaling factor and the crossover rate. It must be emphasized that our goal is not to compete with state-of-the-art continuous optimizers, but rather to provide another proof-of-concept of the possible benefits brought by the AOS paradigm, in a totally different context in relation to both, the underlying EA (just GAs were considered in the previous empirical analyses) and problem domain (continuous in lieu of boolean/combinatorial). Hence, no specific effort was put on tuning the DE parameters with respect to the problem at hand. Population size NP is set to $10 \times d$ (as recommended by [Storn and Price, 2008]), where d denotes the dimension of the search space; mutation scaling factor F is set to .5, and crossover rate CR is set to 1. This latter choice provides to DE the invariance property with respect to rotation, while stressing the impact of the application of the mutation strategies (although being counter-intuitive, $CR = 1$ means no crossover at all, only mutation strategies are applied), consequently emphasizing the gain brought by each AOS scheme on their control.

The set of variation operators is composed of four standard mutation strategies, retaining the same as in [Gong *et al.*, 2010a] for the sake of comparative evaluation:

1. "DE/rand/1": $\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right)$
2. "DE/rand/2": $\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) + F \cdot \left(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}\right)$
3. "DE/rand-to-best/2": $\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot \left(\mathbf{x}_{best} - \mathbf{x}_{r_1}\right) + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) + F \cdot \left(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}\right)$
4. "DE/current-to-rand/1": $\mathbf{v}_i = \mathbf{x}_i + F \cdot \left(\mathbf{x}_{r_1} - \mathbf{x}_i\right) + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right)$

where $\mathbf{x}_i$ is the current (or target) individual, $\mathbf{x}_{best}$ is the current best one, and $\mathbf{x}_{r_1}, \mathbf{x}_{r_2}, \mathbf{x}_{r_3}, \mathbf{x}_{r_4}$ and $\mathbf{x}_{r_5}$ are individuals uniformly drawn in the population.

As mentioned before, two benchmark sets of single-objective continuous functions are available in the BBOB framework, a noiseless [Hansen *et al.*, 2009a] and a noisy [Hansen *et al.*, 2009b] one. Only the noiseless testbed will be considered here. It involves 24 functions divided into 5 classes, according to their most relevant characteristics:

- 5 separable functions;
- 4 functions with low or moderate conditioning;

- 5 unimodal functions with high conditioning;
- 5 multi-modal functions with adequate global structure;
- and 5 multi-modal functions with weak global structure.

Additionally, for each of these 24 functions, there are 15 instances defined by different translation and rotation transformations over the original function. The noiseless testbed, described in detail in [Hansen *et al.*, 2009a], thus totalizes 360 different function instances.

The framework enables experimentation on different dimensions (although the post-processing scripts, by default, consider only $d \in \{2, 3, 5, 10, 20, 40\}$). As a representative set, experiments were done for $d \in \{5, 20\}$. But for the shorter dimension, the results attained are much less interesting: the problems are too quickly solved; consequently, not much significant difference can be observed between the performance of the different AOS schemes. For this reason, only the results for $d = 20$ will be reported here; the results on $d = 5$ can be found in [Fialho and Ros, 2010]. The stopping conditions of each optimization run are: the achievement of the optimum solution f_{opt} (with tolerance 10^{-8}), or the maximum number of function evaluations attained, this latter being fixed at $10^5 \times d$.

An informative measure of performance used in this experimental framework is the so-called Expected Running Time (ERT), which can be defined as follows: given a target function value, ERT is the empirical expected number of function evaluations for achieving a fitness value below the target. Formally, it is measured as the ratio of the number of function evaluations for reaching the target value over successful trials, plus the maximum number of evaluations for unsuccessful trials, divided by the number of successful trials. In addition to the standard ECDF plots used throughout this Chapter, a different kind of plot will be used here, the ECDF-ratio, which clearly depicts the speed-up ratio of one technique with respect to the others.

All the combinations involving rank-based and Extreme-based *Credit Assignment* schemes were tried on this experimental setting and will be compared in the following. Besides, the PM *Operator Selection* technique (Section 4.4.1) will also be considered here, but combined with a different *Credit Assignment* scheme, the average of relative fitness improvements. This combination, proposed in [Gong *et al.*, 2010a] and referred to as PM-AdapSS-DE, is a preliminary outcome of our on-going collaboration with the China University of Geosciences, which later motivated the assessment of the bandit-based approaches on this domain [Fialho *et al.*, 2010b]. For the sake of completeness, the PM-AdapSS-DE method will be reminded in Section 6.6.3.

As done for the other empirical analyses presented in this Chapter, each AOS combination had its hyper-parameters tuned off-line prior to the experiments used to gather the comparative results. The same tuning procedure defined in Section 6.2.2 was used, independently for each dimension. The only difference is that each elimination round happens after one run over all the function instances, what in fact corresponds to 360 performance results for each of the AOS schemes under comparison, up to 11 runs over all instances or one configuration left. The best hyper-parameter configuration found for each of them on dimension 20, used in the experiments that will be analyzed in the following, is presented in Table 6.24.

Credit/OpSel	SLMAB	DMAB	MAB	AP
AbsExt	C100W500	C100G.1W10	C100W500	P.2A.1B.3W500
OpSel/Credit	AUC (Decay)	SR (Decay)	AUC (NDCG)	SR (NDCG)
RMAB (ΔF)	C.5D1W100	C.5D.75W50	C.5W50	C.5W50
RMAB (F)	C.5D.9W50	C.5D.5W50	C.5W50	C.5W50

Table 6.24: Hyper-parameter configurations used on BBOB dimension 20

Besides the comparison between the different AOS methods proposed here, the best AOS combination will be further compared with some other baseline approaches, namely, the Naive uniform selection between the same four strategies, and four variants of DE, each one applying only one of the considered strategies. Finally, a kind of optimal baseline is defined by a state-of-the-art continuous optimizer, the CMA-ES with an Increasing POPulation size restart strategy (IPOP-CMA-ES) [Auger and Hansen, 2005], which was tested with the same parameter tuning as used in [Hansen, 2009a].

It is important to note that, differently from the $\mathcal{ART}$ scenarios (Section 6.3) and the boolean benchmark problems (Section 6.4), in this experimental setting the fitness function should be minimized.

6.6.3 The PM-AdapSS-DE Method

The PM-AdapSS-DE AOS method uses as *Operator Selection* mechanism the Probability Matching (PM), described in Section 4.4.1. Equally motivated by the need of a higher robustness in order to be efficient in a variety of different problems with the same hyper-parameter configuration, its *Credit Assignment* employs a different kind of normalization scheme, that takes place on the impact measurement level; while our Normalized schemes (Section 5.2.3) do so in the *Credit Assignment* output level.

The relative fitness improvement η_i, proposed in [Ong and Keane, 2004], measures the impact of an operator application as:

$$\eta_i = \frac{\delta}{cf_i} \cdot |pf_i - cf_i| \tag{6.1}$$

where $i = \{1, \cdots, NP\}$ refers to each individual of the population of size NP, δ is the fitness of the best-so-far solution in the population, and pf_i and cf_i represent, respectively, the fitnesses of the target parent and of the generated offspring. In case of no improvement (*i.e.*, the offspring is worse than or equal to its target parent), the impact is assessed as being null. Finally, the credit assigned to each strategy is the Absolute Average value of these impact measures.

Besides the relative measure, a main difference of this AOS method with respect to the other methods tried in this thesis is that it assigns credit to the operators and updates their empirical estimates only once per generation, based on their production during the given generation. Hence, there is no hyper-parameter W on the *Credit Assignment* side; the only hyper-parameters that remain to be tuned are the ones from PM: the minimal

probability of selecting each operator p_{min}, and the adaptation rate α. These hyper-parameters were also off-line tuned, in the same way as for the other methods, using the ranges of values defined for AP in Table 6.4: the best configuration found by F-Race uses $p_{min} = 0$ and $\alpha = 0.6$.

Although being a quite simple method, a good performance is achieved mainly due to its robust *Credit Assignment*. However, as discussed in Section 5.2.4, it is still based on the raw values of the fitness improvements to some extent, thus not being as robust as our proposed *Credit Assignment* schemes, as shown in the empirical comparison that will be presented in the following.

6.6.4 Empirical Results

As previously mentioned, a complete set of experiments on this scenario involves 1 run on each of the 15 instances for each of the 24 functions, thus summing up to 360 results for each technique. Given this huge quantity of numerical data, it would be meaningless to present the detailed results for each function in the form of Tables, as done for the previous benchmark scenarios. Standard ECDFs and ECDF speed-up ratio plots are used instead, summarizing the results for the functions altogether, and for each function class.

The main objective of these experiments is to confirm the expectation that, based on the very robust rank-based *Credit Assignment* and on the optimal EvE balance provided by the bandit-based *Operator Selection*, the combinations involving RMAB and the different versions of AUC and SR should perform better than all the other considered methods on such an heterogeneous scenario. In this way, thus, both the performance and the robustness are jointly assessed, as all methods will use a single hyper-parameter configuration over all functions.

This empirical comparison will be performed in four different steps, as follows. Firstly, a representative configuration for the rank-based AOS schemes will be chosen; then it will be compared with the variants of DE using a single mutation strategy; also with other AOS schemes considered in this thesis; and finally with further baseline approaches. Results for each of these steps are summarized, respectively, in Figures 6.16, 6.17, 6.18, and 6.19; they will be now discussed in turn.

Selection of a Representative for the Rank-based Methods

Different alternatives for rank-based *Credit Assignment* were proposed in Sections 5.2.4 and 5.2.5, namely, Area-Under-Curve (AUC) and Sum-of-Ranks (SR), assigning ranks over fitness improvements, and Fitness-based Area-Under-Curve (FAUC) and Fitness-based Sum-of-Ranks (FSR), which are comparison-based methods that use ranks over fitness values. For each of them, there are still two options for the decaying mechanism, the parameterized one, referred to as Decay, and the parameter-less NDCG, which is equivalent to Decay with $d = 0.4$. The total number of possibilities proposed and tried in this thesis, always in combination with the RMAB *Operator Selection* technique, thus sums up to 8. All of them were compared to one another. The ECDF for each of them, for all results over all functions, and separately for each function class, are presented in

Figure 6.16. As can be seen from these aggregated results, their behavior is rather the same; the choice of the method to be compared with the other baseline approaches is thus determined by convenience, as follows.

Firstly, the comparison-based feature provided by the methods that use ranks over the fitness values (FAUC, FSR) can not be precisely assessed on this scenario, as there are no instances being defined by monotonous transformations over the original function [Hansen *et al.*, 2009a]. Therefore, it becomes interesting to note that, although providing such extra level of robustness, these methods still perform equivalently to the other versions that use ranks over fitness improvements (AUC and SR), while in the previous experimental scenarios the latter frequently outperformed the former. Hence, it is preferred to keep the comparison-based ones. Then, between FAUC and FSR, the first is chosen, by having shown a better performance on a couple of scenarios, although being significantly equivalent in most cases. Finally, the NDCG version is preferred, by being parameter-less. The chosen rank-based AOS combination is thus NDCG/FAUC-RMAB, which will be referred to as FAUC-B in the following, for the sake of brevity.

Comparison of FAUC-B with DE using a Single Strategy

FAUC-B is firstly compared with the standard DE algorithm using a single strategy, with four variants ($DE1 \dots DE4$), each one using one of the four mutation strategies considered by the AOS schemes, defined in Section 6.6.2.

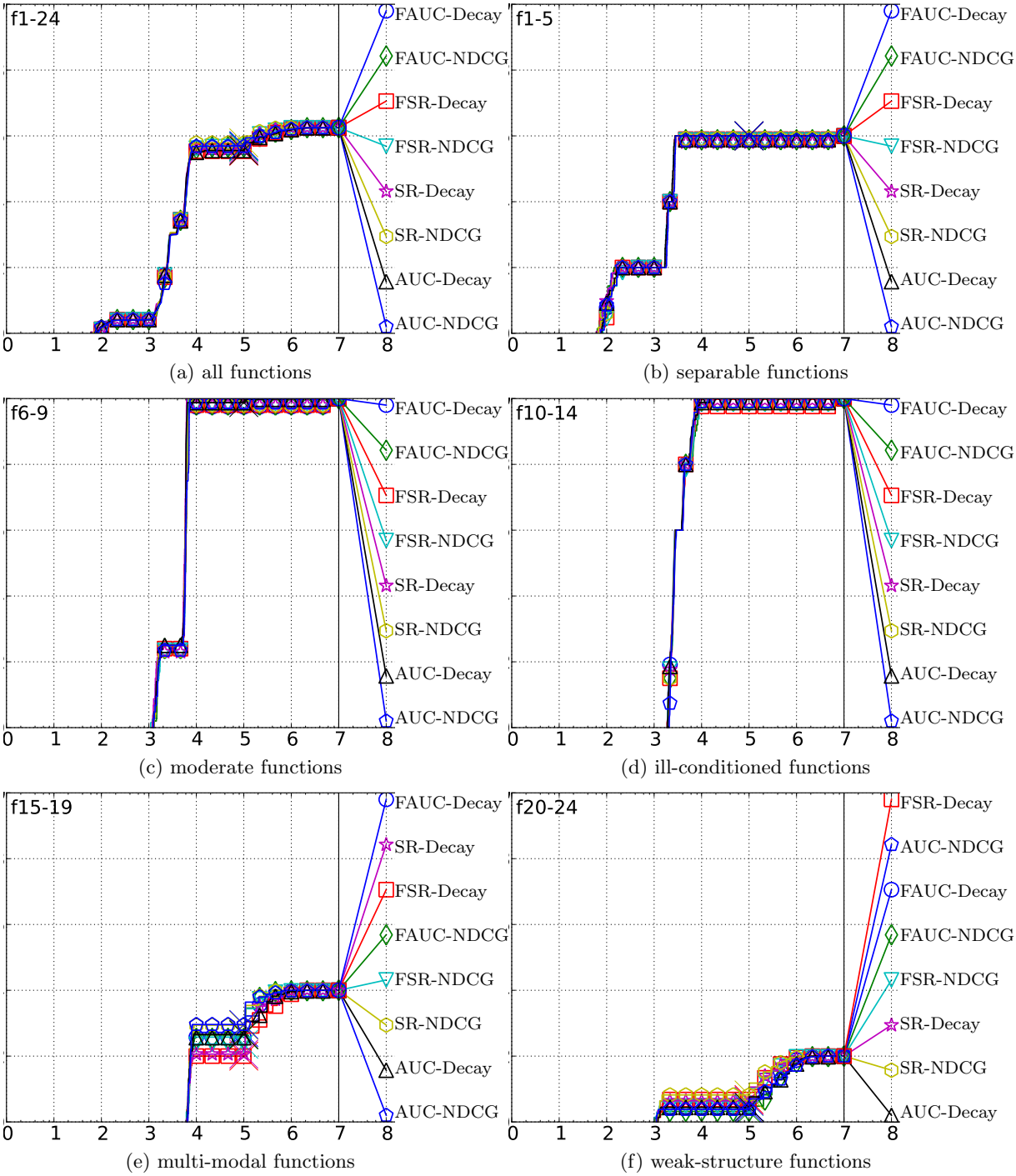
The global picture (Figure 6.17a) shows in the legends that FAUC-B, DE1 and DE3 are able to solve at least one instance for 15 functions out of 24, while DE2 solves 12. DE4 shows unable to solve any of the functions on this dimension (although being occasionally used by the adaptive schemes); hence, it will be neglected in the remainder of this analysis.

Compared with $DE1$, FAUC-B shows to be around 3 times faster in approximately 90% of all the considered cases. $DE2$ is around 20 times slower than FAUC-B on around 65%, 50%, 80% and 40% of the trials, respectively, for the *separable*, *moderate*, *ill-conditioned* and *weak-structure* function classes, while not solving any instance for the *multi-modal* class. $DE3$ is the best one out of the single strategies, performing roughly 10 times faster than $DE2$; overall, it is around 2 times slower than FAUC-B.

It is worth noting that all the schemes failed on most multi-modal functions: 2 out of the 5 *separable* functions are multi-modal and were not solved by any of the schemes, in addition to the *multi-modal* class, in which only 2 out of 5 functions were solved by some of the schemes, and the *weak-structure* class, where only 1 out of 5 functions were solved in at least one trial.

Comparison of FAUC-B with other AOS Schemes

The second series of experiments compare FAUC-B with the other bandit-based approaches proposed in this thesis, namely, MAB (Section 5.3.1), DMAB (Section 5.3.2), and SLMAB (Section 5.3.3), and with the baseline AP (Section 4.4.2), all using the Absolute Extreme *Credit Assignment* (Section 5.2.2). It is also compared with PM-AdapSS-DE, the method that uses PM being fed by Absolute Average of relative fitness improvements, detailed in

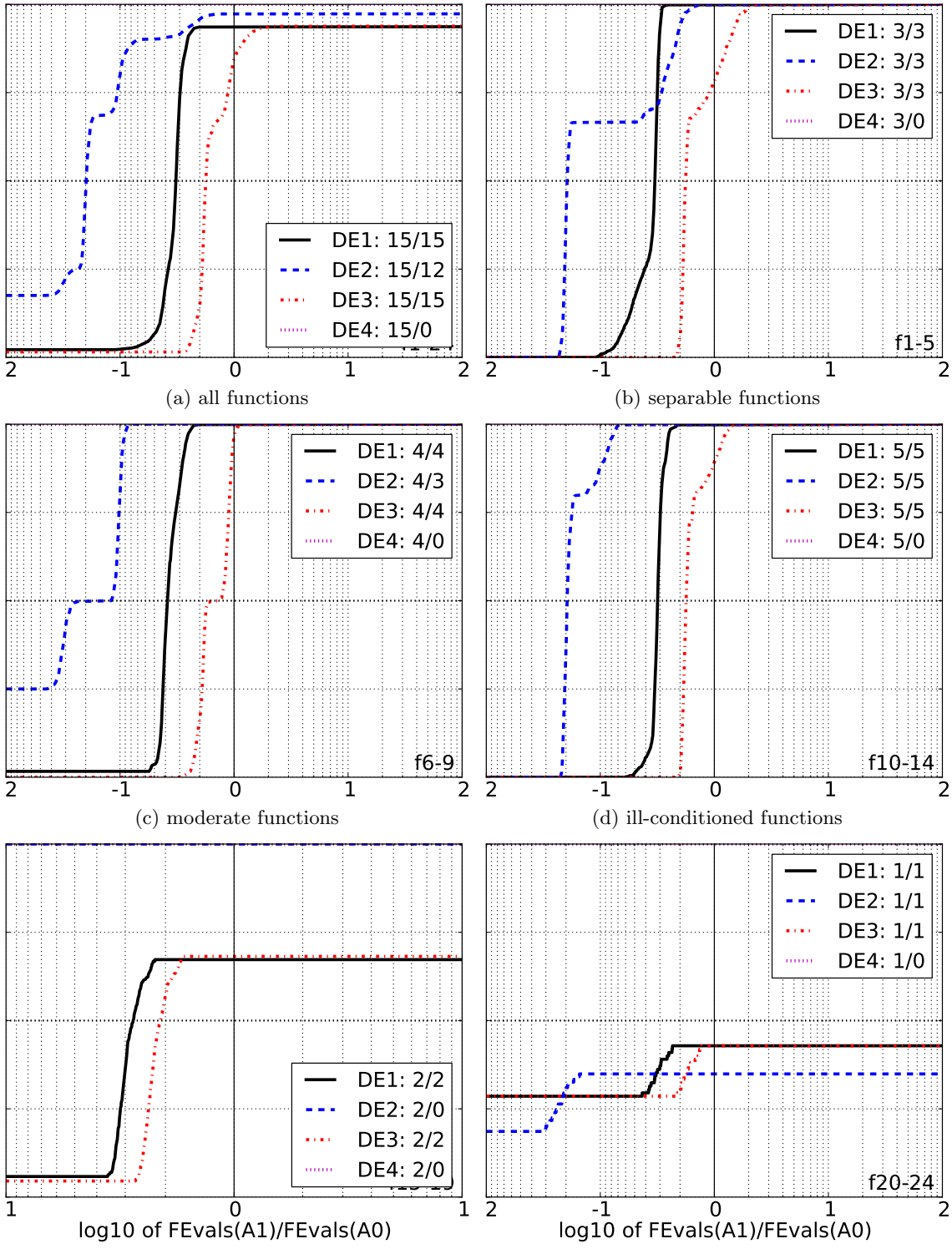


(a) all functions (b) separable functions

(c) moderate functions (d) ill-conditioned functions

(e) multi-modal functions (f) weak-structure functions

Figure 6.16: Standard ECDF plots of the distribution of ERT over dimension for combinations of RMAB with all the rank-based credit assignment schemes, for all functions and sub-groups with $d = 20$ and target 10^{-8}.



(a) all functions (b) separable functions

(c) moderate functions (d) ill-conditioned functions

(e) multi-modal functions (f) weak-structure functions

Figure 6.17: ECDFs of the speed-up ratios in dimension $d = 20$ for the FAUC-B compared with the DE using only one out of the four available mutation strategies. The speed-up ratios are the pairwise ratios of the number of function evaluations for FAUC-B to surpass the target function value 10^{-8}, over the one of the baseline techniques, over all trials for each function. Pairs where both trials failed are disregarded, pairs where one trial failed are visible in the limits being > 0 or < 1 (for this reason, some lines are not visible, as they coincide with the axes). The legends also indicate the number of functions that were solved in at least one trial (FAUC-B first).

Section 6.6.3.

Globally speaking (Figure 6.18a), FAUC-B is approximately 1.5 times faster than most of the other AOS schemes in around 90% of the trials, except for the PM-AdapSS-DE, which is outperformed in approximately half of the trials, performing faster than FAUC-B in around 15% of the function trials. More specifically, FAUC-B is up to 10 times faster than the standard MAB in half of the trials, while being 2 times faster than DMAB and SLMAB in around 75% of the trials.

This global assessment corresponds roughly to the speed-up ratios attained on the *separable*, *moderate* and *ill-conditioned* function classes. Again, none of the schemes are able to perform well on multi-modal functions, with only 2 functions being solved by all schemes in the *multi-modal* class, and only 1 in the harder *weak-structure* class. Although the low number of successful trials on these latter classes, FAUC-B still outperforms all the schemes in most trials.

Comparison of FAUC-B with further Baseline Approaches

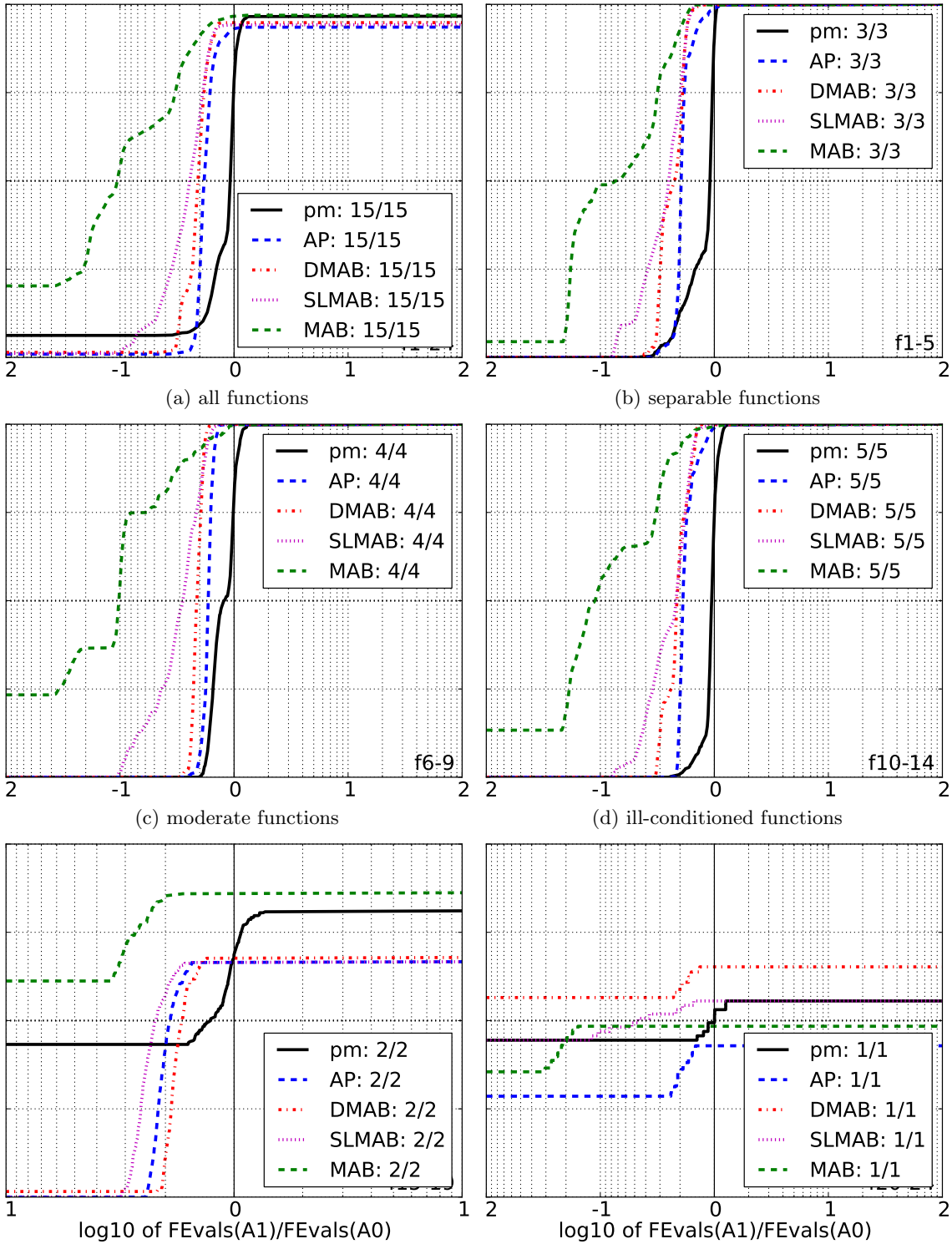
The last empirical comparison, whose results are illustrated in Figure 6.19, considers two extremes. On the one side there is the Naive uniform operator selection. FAUC-B shows to be around 1.5 times faster than *Naive-DE* on around 80% of the trials for all unimodal functions in the *separable*, *moderate* and *ill-conditioned* function classes. It attains the same speed-up ratio on approximately 65% and 55% of the trials, respectively, for the *multi-modal* and *weak-structure* function classes.

On the other side there is the state-of-the-art continuous optimizer IPOP-CMA-ES [Auger and Hansen, 2005], which significantly outperforms FAUC-B in around 90% of the trials for all cases. Besides, it succeeds in solving all the 5 functions for the *multi-modal* class, while FAUC-B and all the other schemes previously considered solve only 2; and it also solves 2 functions for the *weak structure* class, one more than all the other schemes.

6.6.5 Discussion

Differently from the other scenarios previously analyzed in this Chapter, the use of the BBOB experimental framework enabled a much broader and realistic assessment of the proposed AOS schemes. By the evaluation of the methods on the many functions with different characteristics and levels of difficulty, treated as black-boxes, it showed to be an ideal scenario in order to depict the gain in robustness brought by the use of rank-based *Credit Assignment* schemes.

And indeed, the NDCG/FAUC-RMAB (simply referred to as FAUC-B) AOS method, a representative of all the proposed rank-based methods, succeeded in outperforming in the vast majority of the cases: the standard DE using each of the considered mutation strategies, the Naive uniform strategy selection, all the other bandit-based approaches, as well as two other baseline adaptive schemes, Adaptive Pursuit and PM-AdapSS-DE. This performance improvement of the FAUC-B with respect to the others, in terms of Expected Running Time (ERT) to achieve a given function target value, is attributed mostly to: (i) the use of a rank-based *Credit Assignment*, which is robust with respect to

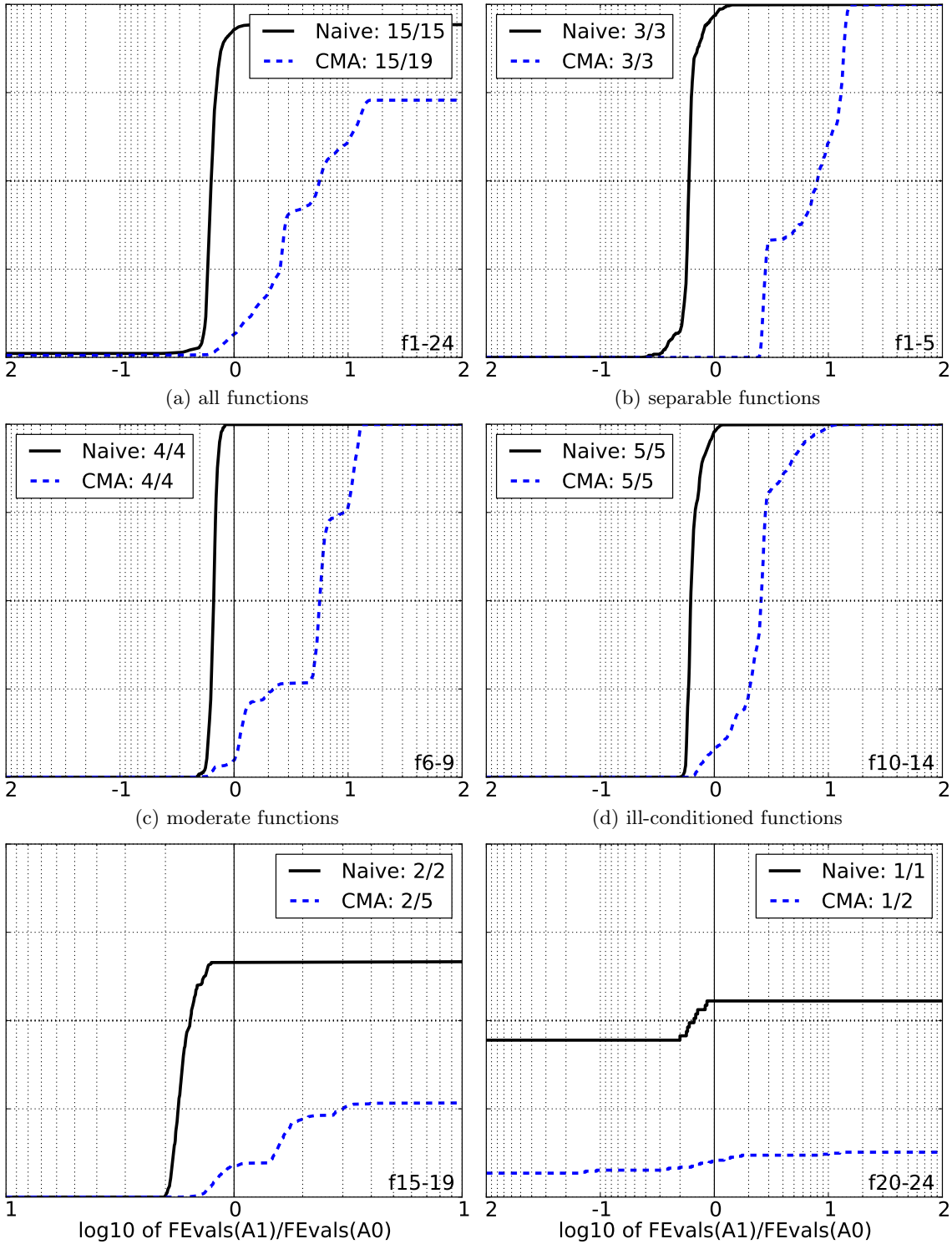


(a) all functions (b) separable functions

(c) moderate functions (d) ill-conditioned functions

(e) multi-modal functions (f) weak-structure functions

Figure 6.18: ECDFs of the speed-up ratios in dimension $d = 20$ for the FAUC-B compared with other AOS combinations. The speed-up ratios are the pairwise ratios of the number of function evaluations for FAUC-B to surpass the target function value 10^{-8}, over the one of the baseline techniques, over all trials for each function. The legends indicate the number of functions that were solved in at least one trial (FAUC-B first).



(a) all functions (b) separable functions

(c) moderate functions (d) ill-conditioned functions

(e) multi-modal functions (f) weak-structure functions

Figure 6.19: ECDFs of the speed-up ratios in dimension $d = 20$ for the FAUC-B compared with the Naive uniform operator selection and the state-of-the-art IPOP-CMA-ES optimizer. The speed-up ratios are the pairwise ratios of the number of function evaluations for FAUC-B to surpass the target function value 10^{-8}, over the one of the baseline techniques, over all trials for each function. The legends also indicate the number of functions that were solved in at least one trial (FAUC-B first).

the very different situations tackled within this benchmark suite, while efficiently following the changes in the qualities of the strategies (the reduction of the AUC for one operator, by definition, results in the augmentation of the AUC for one of the others); and to (ii) the use of a bandit-based *Operator Selection*, which has already shown to be very efficient in the other experimental settings. FAUC-B, as well as the other methods using ranks over the fitness values instead of fitness improvements, still provide an extra layer of robustness that was not challenged in this scenario, that of being comparison-based: in case there were instances derived by monotonous transformations over the original functions, the performance gap could be even bigger, possibly with the comparison-based methods (FAUC and FSR) significantly outperforming the simply rank-based methods (AUC and SR). This situation should be further addressed in the near future.

Moreover, PM-AdapSS-DE [Gong *et al.*, 2010a] (Section 6.6.3), showed to be the best out of the four other adaptive schemes, what confirms the gain in robustness achieved by the use of a relative instead of a raw reward. It is also worth noticing that, as for the other simpler benchmark scenarios, DMAB greatly outperformed the standard MAB by the use of its change-detection test, while SLMAB was able to closely follow its performance, both of them performing very similarly to AP. But, although improving over MAB, both DMAB, SLMAB and AP techniques performed rather equivalently to the Naive uniform strategy, and this puts into question their efficiency in this experimental setting.

Compared to the Naive uniform approach, it is true that the efficiency gain presented by FAUC-B seems to be moderate in relation to the price to pay for an adaptive scheme in terms of computational complexity. Nevertheless, it should be observed that the Naive uniform strategy considers here a small number of strategies, most of which performing well: $DE1$ and $DE3$ perform quite well; although much slower, $DE2$ still reaches the target; the only inefficient strategy is $DE4$. In the general case, however, the performance of the strategies is unknown; the performance of the above strategies was assessed through extensive experiments. The use of an AOS scheme remains thus relevant in the general case.

Additionally, though much improved over the results of all mentioned approaches used within DE, the best results of the FAUC-B +DE algorithm remains far below those of the state-of-the-art optimizer IPOP-CMA-ES. But the DE algorithm that FAUC-B has been applied to is the very standard one; several improvements have been recently proposed to it, *e.g.*, adding adaptive parameter control for F and CR [Qin *et al.*, 2009]. The applicability of FAUC-B in DE framework opens the path for fruitful research using the numerous recent DE variants.

Another further work is to address the multi-modality issue: all tested algorithms failed on most multi-modal functions (40% of the *separable* class, plus all the functions for the *multi-modal* and *weak structure* classes). In the same way as for the work on SAT problems (Section 6.5), in order to efficiently tackle multi-modal problems, the maintenance of some level of diversity in the population should also be accounted somehow for the rewarding of operator applications, as discussed in Section 4.5.2.

6.7 Hyper-Parameters Analysis

As discussed in Section 6.2.1, the AOS mechanisms analyzed in this work have their own hyper-parameters (Table 6.2), whose values might critically impact their performances. This issue has been addressed using the F-Race, as described in Section 6.2.2, and the best hyper-parameter configuration for each considered AOS has been determined and further used on each of the different empirical comparisons presented in this Chapter. It is, however, important to study the hyper-parameters, in order to identify which of them should receive more attention during the tuning process when addressing a new problem. An analysis of their *sensitivity* will be presented in Section 6.7.1.

Another important aspect concerning the hyper-parameter configuration of the AOS schemes is how robust they are when tackling different problems. For instance, the methods based on the raw values of fitness improvements are expected to be very problem-dependent, as discussed in Section 5.2.3, while the rank-based methods are expected to be more robust to different situations. An analysis of their *robustness* will be presented in Section 6.7.2.

The results that will be analyzed here were partially published in [Fialho *et al.*, 2010a; Fialho *et al.*, 2010c; Fialho and Ros, 2010].

6.7.1 On the Sensitivity of the Hyper-Parameters

When only 1 or 2 hyper-parameters are concerned, a 3-D plot of the response surface of these parameters gives a clear picture (as has been done in [Da Costa *et al.*, 2008] for AP and DMAB on some artificial scenarios, for instance). However, for each AOS technique, there are the hyper-parameters of both *Operator Selection* and *Credit Assignment* schemes: only MAB and SLMAB combined with the Extreme or Average *Credit Assignment* schemes have 2 hyper-parameters; DMAB and AP, combined with the same *Credit Assignment* schemes, have respectively 3 and 4 hyper-parameters; and RMAB with any of the rank-based schemes has 3 (see the list in Table 6.2). This is why ECDF plots will be also used for this analysis, as described in the following. Another alternative could have been to use some parameter setting procedure like REVAC [Nannen and Eiben, 2007] (Section 3.3.2), that gives an idea of the sensitivity of all parameters it optimizes while finding its optimal value.

ECDF Sensitivity Plots

Though ECDF sensitivity plots have been generated for all AOS schemes and all scenarios previously presented, only a few series of plots offering typical behavior will be depicted and analyzed into detail here. All the non rank-based methods are considered with the AbsExt *Credit Assignment*. This scheme was chosen because it performs best in most cases; for a few exceptions, the *Instantaneous* performed slightly better, but the corresponding ECDF plots looked rather similar, though involving one less hyper-parameter, the size W of the rewards window (expect for SLMAB, which needs W anyway). For the rank-based methods, the RMAB with the Decay/AUC *Credit Assignment* is considered. Accordingly,

the other alternatives could have been tried, but they present rather similar performance in most scenarios; between Decay and NDCG, the former is preferred so as to be able to also analyze the decay factor D. For each of these AOS combinations, its sensitivity with respect to its hyper-parameters will be analyzed on four benchmark problems, a kind of representative set of all the different problems considered in this thesis: the Uniform scenario with $\Delta T = 500$, the $\mathcal{ART}(0.1, 39, 0.5, 3)$ with $\Delta T = 200$, and the OneMax and Royal Road boolean benchmark problems.

All figures display a number of ECDF curves representing the results of the same AOS method on the same scenario. For the artificial scenarios, the x-axis represents the TCR, ranging from the value of the optimal strategy (*i.e.*, the highest possible value on average for any AOS) **down to** the value that would, on average, be gathered by the Naive uniform strategy (hopefully the lowest possible value on average for any AOS). For the OneMax and Royal Road cases, the x axis represents the number of generations to achieve the optimum, starting from 4500 for the former and zero for the latter problem, **up to** the average number of generations taken by the Naive uniform choice. The y-axis shows the proportion of runs that reached the corresponding x value, out of the total number of runs considered.

For each benchmark problem, a large amount of experimental results have been gathered during the Racing procedure, and at least 11 runs have been performed for all hyper-parameter combinations of the factorial design sketched by the values in Table 6.4. The results over these 11 runs will be used for the analysis on the OneMax and Royal Road problems; this explains possible divergences between the best configurations considered here and the ones presented in the tables of results corresponding to each scenario. Experiments on the Uniform and $\mathcal{ART}$ artificial scenarios are significantly cheaper, in terms of computational time; hence, for these scenarios, even the parameter configurations that did not make it through the end of the Racing procedure have been run 50 times for the sake of the sensitivity analysis.

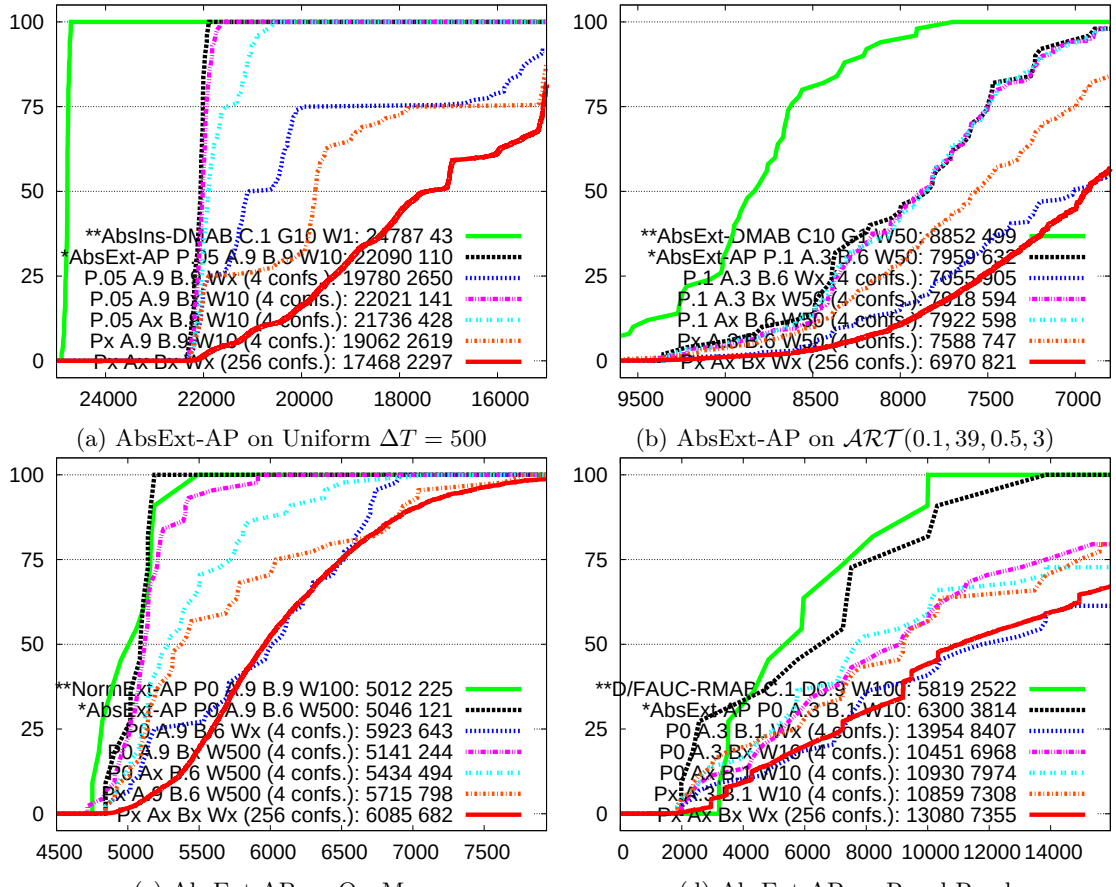
Two lines are given as references on each plot: the top-left-most line (continuous, and green on color printouts), labelled with a "**" and referred to as "Best/All", represents the overall best results, in terms of average TCR (or number of generations to the optimum in the OneMax and Royal Road cases), obtained on this scenario by a single hyper-parameter configuration, between all the AOS combinations, after the number of runs considered for each scenario. The next line going down/right, labelled with a "*" and referred to as "Best/This", represents the results obtained by the best configuration of hyper-parameters for the AOS scheme named on the caption of the sub-figure, under the plot. The discrepancy between both lines shows in detail how different are the performances of the considered AOS and of that of the method that performed best on this scenario.

At the other extreme of each sub-figure, the bottom-right-most line (solid, red on color printouts) represents the ECDF of all runs for the particular AOS, *i.e.*, for all hyper-parameter configurations ever tried (the average performance of the complete factorial design from values given on Table 6.4). The lines in-between represent partial aggregations of these runs. More precisely, if the best results have been obtained for a given set of hyper-parameters (recalled in the legend), each line represents the ECDF obtained when one

hyper-parameter is varied over all its values used in the Racing procedure, while all others are kept to the optimal value found. If a line corresponding to a given parameter is close to the line of the best configuration, it is a clear indication that the results are not very sensitive to this hyper-parameter. Oppositely, a high difference indicates a high sensitivity. Besides the ECDF curve, the average and standard deviation of the performance are also presented in the legends of the plots for each of the different aggregations considered.

Sensitivity Analysis

In order to facilitate the comparison of the impact of the hyper-parameters on the performance of each AOS technique on the different benchmark problems, and possibly find common sensitivity hints for their tuning, the plots are grouped by technique. Figure 6.20 presents the ECDF sensitivity plots for the baseline AbsExt-AP method, for the sake of comparison. Figure 6.21 depicts in each column the plots for both AbsExt-MAB and AbsExt-SLMAB, while Figure 6.22 shows the plots for AbsExt-DMAB on the right and Decay/AUC-RMAB on the left column.



(a) AbsExt-AP on Uniform $\Delta T = 500$

(b) AbsExt-AP on $\mathcal{ART}(0.1, 39, 0.5, 3)$

(c) AbsExt-AP on OneMax

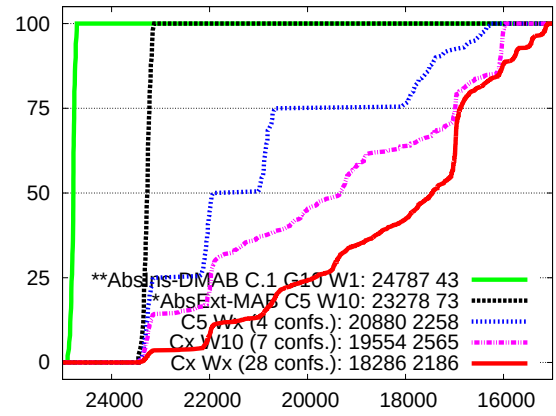
(d) AbsExt-AP on Royal Road

Figure 6.20: ECDF sensitivity plots for AbsExt-AP

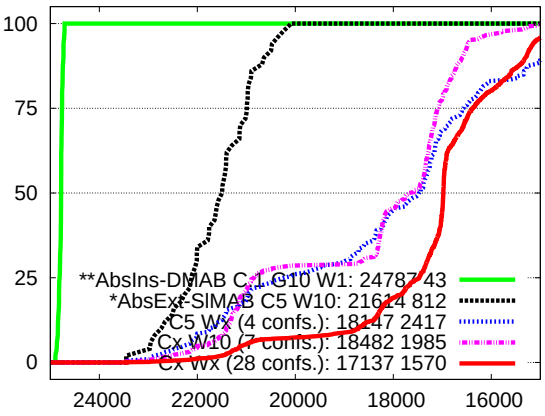
Starting with the combination of Adaptive Pursuit (AP) *Operator Selection* and Absolute Extreme (AbsExt) *Credit Assignment*, the global picture with respect to sensitivity is rather clear. The adaptation rate α and the learning rate β are very robust indeed: their aggregated ECDF plots are very close to the curves representing the best configuration for this technique on all problems. The minimal probability p_{min} is a much more sensitive parameter, as expected (and as discussed in Section 4.4.2); but its sensitivity clearly (and intuitively) depends on the number of operators and on the maximum value tried for it: on the $\mathcal{ART}$ scenario it seems to be as insensitive as α and β (Figure 6.20b); however, only two operators are considered in this case, thus the total exploration when using the maximum value tried for p_{min} (= 0.2) sums up to "only" 40% of exploration; while on the Royal Road scenario, which has 5 operators, p_{min}=0.2 refers to a complete Naive uniform behavior. The window size W is the most sensitive parameter on all cases. Altogether, AP seems to be quite robust with respect to its hyper-parameters, but its global performance, when compared to the bandit-based approaches on most benchmark scenarios considered in this thesis, make it a poor choice anyway.

The scaling factor C and the sliding window size W are common hyper-parameters for all bandit-based approaches. Starting with C, for MAB, SLMAB and DMAB, it is definitely a very sensitive parameter. As discussed in Section 5.3.4, when using *Credit Assignment* schemes based on the raw values of fitness improvements, C has a double role on the bandit-based approaches: besides controlling the Exploration versus Exploitation balance of the *Operator Selection*, it needs to account for the different ranges of fitness improvements (which tend to vary as the search goes on, and according to the problem at hand). For RMAB, C still seems to be a sensitive hyper-parameter; however, as can be seen in the captions of the respective plots, the winner configurations for the different problems all use $C \leq 1$, while values up to 100 are being considered in the corresponding (dark blue) curve, consequently greatly degrading its aggregated performance. From this alternative analysis, thus, it can be said that the use of a rank-based scheme fulfills its original motivation, that of providing an invariant range of rewards during all the search process on a given problem, and over different problems, what reflects in similar C values being used by the winner configurations. Conversely, the winner configurations of the other bandit-based approaches use very different values for C on the different problems; for instance, SLMAB uses $C = 0.01$ on the OneMax problem, and $C = 100$ on the $\mathcal{ART}$ problem. More on this will be discussed in the robustness analysis presented in Section 6.7.2.

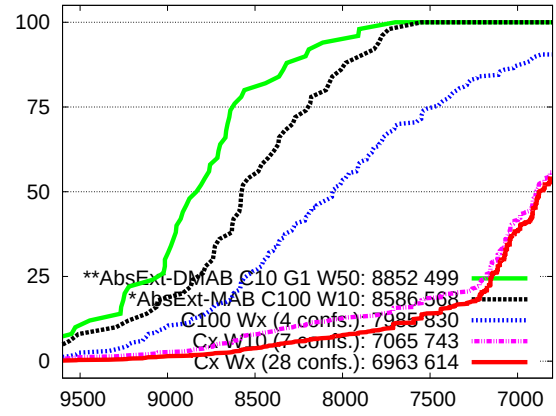
The window size W also seems to be very sensitive for the bandit-based approaches, specially on the Uniform and $\mathcal{ART}$ scenarios, in which there is a strong link between W and ΔT: for values of W larger than the epoch size, the frequency of the changes will result in using too old information. This issue becomes even more important for the approaches other than RMAB, in which there is one window for each operator: in the worst case, a window might contain rewards as old as $K \times W$ steps ago, K being the number of operators. For instance, the "steps" shown in the corresponding ECDF curves for MAB, DMAB and RMAB on the Uniform scenario (respectively, on Figures 6.21a, 6.22a and 6.22b), clearly depict how large values for W are hindering the overall performance of the aggregated distribution in this case. This hyper-parameter becomes less sensitive in



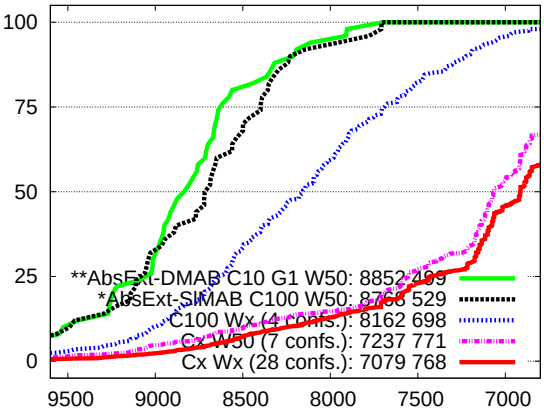
(a) AbsExt-MAB on Uniform $\Delta T = 500$



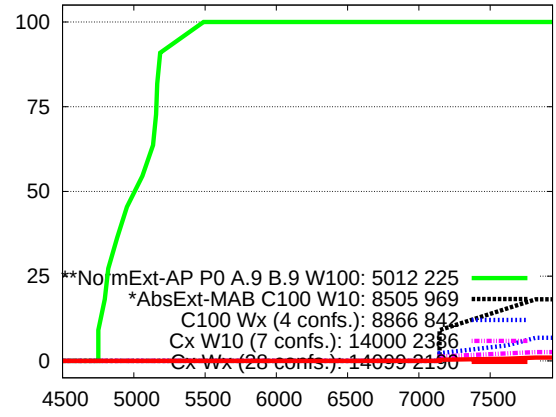
(b) AbsExt-SLMAB on Uniform $\Delta T = 500$



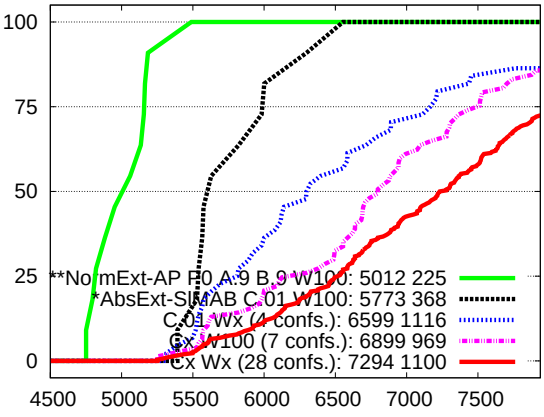
(c) AbsExt-MAB on $\mathcal{ART}(0.1, 39, 0.5, 3)$



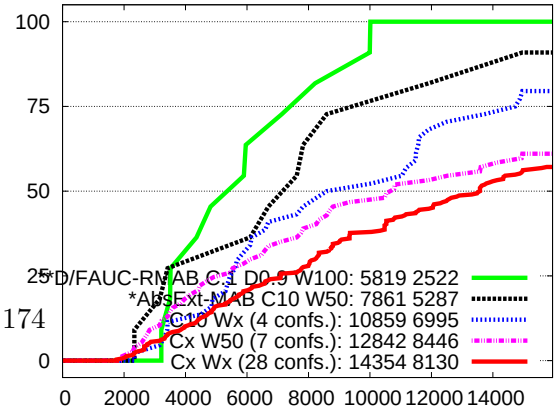
(d) AbsExt-SLMAB on $\mathcal{ART}(0.1, 39, 0.5, 3)$



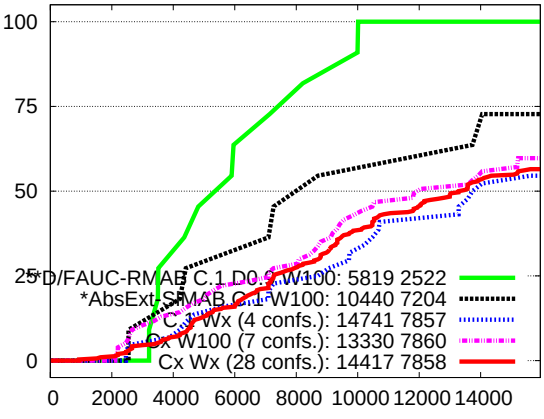
(e) AbsExt-MAB on OneMax



(f) AbsExt-SLMAB on OneMax



(g) AbsExt-MAB on Royal Road



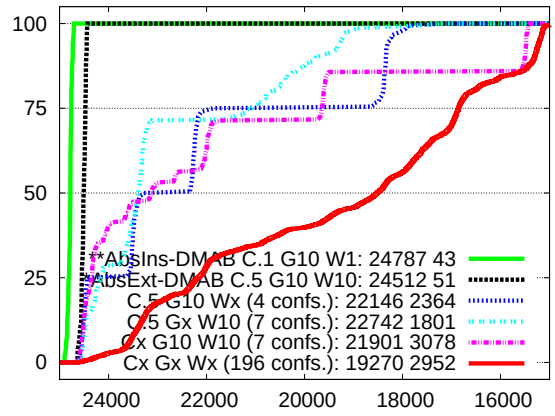
(h) AbsExt-SLMAB on Royal Road

Figure 6.21: ECDF sensitivity plots for AbsExt-MAB and AbsExt-SLMAB

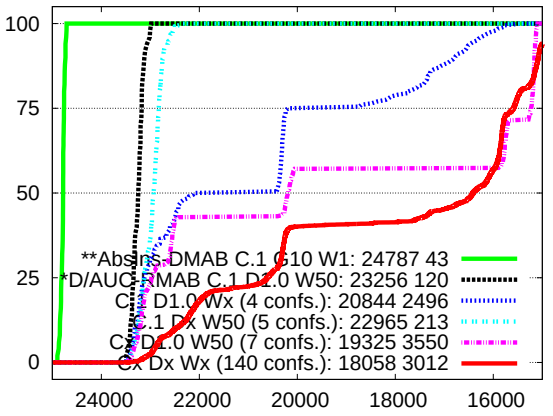
the more realistic OneMax and Royal Road scenarios, in which there is no clear relation between operators qualities and time. For SLMAB, the W hyper-parameter has a double role: it controls both the size of the sliding window and the size of the intrinsic memory of the relaxation update rule used on the *Operator Selection* side; for this reason, it seems to be much more sensitive to this hyper-parameter than the other bandit-based approaches on three out of the four cases (except for the $\mathcal{ART}$, in which it is as sensitive as for the other approaches). More specifically, in the case of RMAB, as there is only one window for all operators, (intuitively) larger window sizes are preferred.

The DMAB change-detection threshold γ shows not to be sensitive on the Uniform and $\mathcal{ART}$ scenarios. A tentative interpretation goes as follows. As the range of rewards for the best operator are the same during the whole search process, on the Uniform scenario it seems that 3/4 of values tried for DMAB γ perform well (see again the "step" on around 75% of the cyan-colored distribution in Figure 6.22a); indeed, in this case, from the values explored, only $\lambda \in \{100, 1000\}$ achieve a TCR much worse than the others, as these thresholds are too big in relation to the actual changes in the reward distributions, thus not triggering any restart. Along the same lines, on the $\mathcal{ART}$ scenario, whenever there is an exchange in the reward distributions, *i.e.*, whenever an outlier reward (of value 39 in this case) is received, as it is much higher than the expectation of the previous distribution (= 3), most values tried will succeed in detecting the change. For the OneMax and Royal Road problems, the best operator might present very different expected reward during the search process, thus there is no value for γ that works optimally during all the search; for this reason, in these cases, the γ threshold appears to be as sensitive as the very sensitive scaling factor C. In more realistic scenarios, having different reward ranges during the search is very likely to be the case; hence, this hyper-parameter tends to hinder the performance of DMAB (although still outperforming the standard MAB in most cases), while requiring a considerable amount of computational time for its off-line tuning when compared to the other methods.

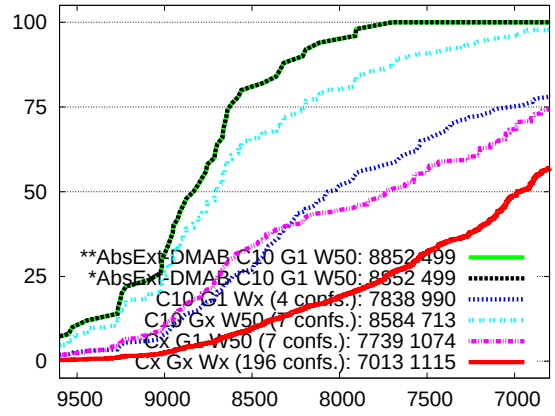
Accordingly, the RMAB ranking decay factor D is also much less sensitive on the Uniform scenario: as the rewards received are always already ranked somehow according to the quality of the operator (despite some overlap between subsequent operators), the operator qualities are already ranked by construction, thus not needing an extra decaying factor in order to efficiently differ between them. Anyway, even for the other scenarios, it does not seem to be as sensitive as the other hyper-parameters; expect for the Royal Road case, in which it is surprisingly more sensitive than W, although much less sensitive than the scaling factor C. Another empirical proof of the non-sensitivity of this hyper-parameter is related to the fact that, in most empirical performance comparisons presented in this Chapter, the Decay version using some high value for D showed equivalent performance to the $NDCG$ version, which is basically the same as using Decay with $D = 0.4$, as shown in Figure 5.1. It is also worth noticing that the fact that this hyper-parameter is always bounded between 0 and 1 results in a much easier and cheaper off-line tuning than those of the scaling factor C and the change-detection threshold γ, which have no known bounds.



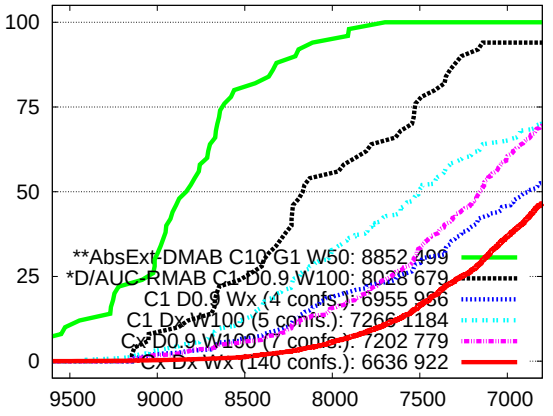
(a) AbsExt-DMAB on Uniform $\Delta T = 500$



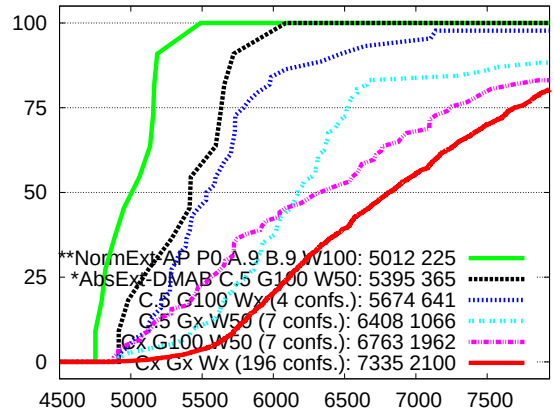
(b) Decay/AUC-RMAB on Uniform $\Delta T = 500$



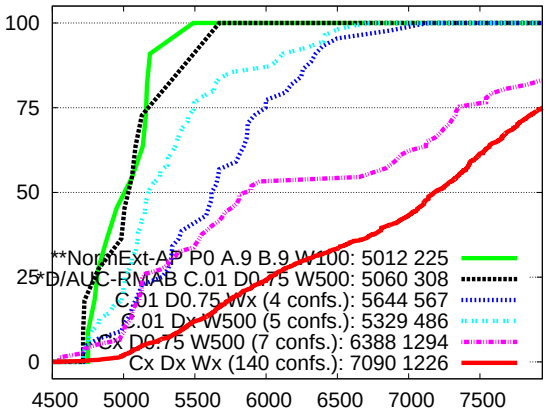
(c) AbsExt-DMAB on $\mathcal{ART}(0.1, 39, 0.5, 3)$



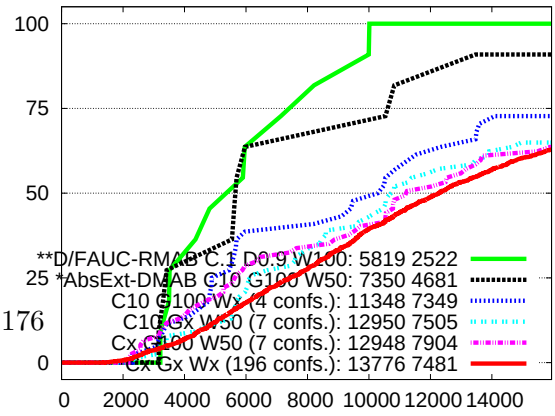
(d) Decay/AUC-RMAB on $\mathcal{ART}(0.1, 39, 0.5, 3)$



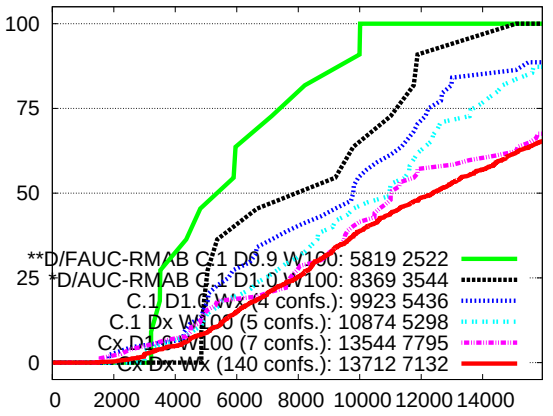
(e) AbsExt-DMAB on OneMax



(f) Decay/AUC-RMAB on OneMax



(g) AbsExt-DMAB on Royal Road



(h) Decay/AUC-RMAB on Royal Road

Figure 6.22: ECDF sensitivity plots for AbsExt-DMAB and Decay/AUC-RMAB

6.7.2 On the Robustness of the Hyper-Parameters

The sensitivity of a given hyper-parameter might be alleviated by the robustness of the method with respect to many different problems, *i.e.*, even if a lot of effort is needed for the preliminary off-line tuning of this parameter, in case the configuration found by the off-line tuning performs reasonably well for several different problems, the computational budget spent on this initial effort might become worth the expense.

In this Section, the robustness of the AOS schemes with respect to their hyper-parameters is analyzed on two experimental scenarios: firstly, on the OneMax problem and on 3 other problems defined by monotonous transformations of OneMax; then, on the very heterogeneous set of functions provided by BBOB.

Robustness on Transformations over the OneMax Problem

A first series of experiments was conducted, based on the OneMax problem, to analyze the expected gain in robustness provided by the use of the rank-based *Credit Assignment* schemes (Section 5.2.4) and, specially, the invariance with respect to monotonous transformations (leading to the so-called comparison-based property) featured by the schemes using ranks over the fitness values instead of ranks over the fitness improvements (Section 5.2.5).

As presented in Section 6.4.2 (and as done for every performance benchmarking scenario considered in this Chapter), the F-Race procedure was first used to tune the hyper-parameters of all the AOS schemes prior to the comparison of their average empirical performance on the OneMax problem. For the robustness analysis that will be presented here, this same hyper-parameter configuration found to be the best on the OneMax problem for each technique was used for its assessment on functions defined by monotonous non-linear transformations over the original OneMax function $\mathcal{F}$, namely: $\log(\mathcal{F})$, $\exp(\mathcal{F})$, and $\mathcal{F}^2$.

The complete results, gathered with the same experimental setting used in the original performance comparison presented in Section 6.4.2, can be found on Table 6.25. The average performance (number of generations to optimum) on the original and on the three transformed functions are presented for each AOS scheme. These performance measures are ranked independently for each function, and the first column of the Table summarizes this ranks, by presenting their sum; the Table is sorted by this column: the more robust technique is the one with lowest values for $\sum r$. Additionally, the second column presents the gap between the worst and the best performance achieved by each AOS scheme over all four functions, what can be seen as a complementary view of their robustness with respect to this experimental scenario. Finally, as done on the previously analyzed performance comparisons, the best result for each function is highlighted with bold-face and grey background, like **this**, and the statistically equivalent results are displayed with a grey background.

These results empirically confirm several expectations. Firstly, the rank-based methods that use the rank of the fitness improvements, AUC-RMAB and SR-RMAB, achieve an overall better performance, while showing to be quite robust with respect to the

$\sum r$	(h-l)	$\mathcal{F} = \sum b_i$	$\log(\mathcal{F})$	$\exp(\mathcal{F})$	$(\mathcal{F}^2)$	AOS technique
12	485	5103 427	**5195 430**	5562 950	5588 950	Decay/AUC-RMAB
13	244	5231 503	5421 524	5475 422	5475 422	NDCG/SR-RMAB
15	362	5215 374	5347 547	5577 634	5575 534	Decay/SR-RMAB
18	1321	**5097 230**	6295 1176	**4974 201**	**4984 184**	NormExt-AP
19	108	5444 252	5458 382	5511 347	5403 332	NormIns-DMAB
24	300	5366 478	5434 596	5650 887	5666 881	NDCG/AUC-RMAB
31	**0**	5652 644	5652 644	5652 644	5652 644	Decay/FSR-RMAB
31	807	5123 218	5431 223	5930 334	5792 382	AbsExt-AP
36	**0**	5667 729	5667 729	5667 729	5667 729	NDCG/FSR-RMAB
41	**0**	5726 399	5726 399	5726 399	5726 399	Decay/FAUC-RMAB
46	54	5728 204	5767 312	5780 329	5726 263	NormIns-AP
50	172	5718 239	5805 279	5748 227	5890 386	AbsIns-AP
54	**0**	5796 420	5796 420	5796 420	5796 420	NDCG/FAUC-RMAB
54	186	5750 251	5782 251	5936 277	5875 287	AbsAvg-AP
62	355	5790 226	5910 271	6048 337	6145 319	NormAvg-AP
65	2591	5376 285	7967 718	7722 2151	6138 516	AbsExt-DMAB
74	706	6427 597	6956 784	7133 866	6808 691	NormExt-SLMAB
77	3310	5508 823	8818 3653	7173 3288	6865 2861	NormExt-DMAB
80	2673	5480 276	8079 743	7961 653	8153 684	AbsIns-DMAB
80	502	7193 1614	6890 1294	7370 1033	7392 1117	NormAvg-DMAB
81	2285	5997 593	6345 644	8282 2044	7639 1807	NormExt-MAB
83	1905	6662 961	8277 1553	6507 839	6372 786	NormIns-SLMAB
92	225	8013 671	7893 681	8021 582	8118 738	NormIns-MAB
96	365	7903 638	8023 716	8268 949	8008 874	NormAvg-MAB
97	631	7494 611	8122 724	8125 1159	8075 740	AbsAvg-DMAB
108	6971	6059 667	8863 694	13030 3053	12136 949	AbsExt-SLMAB
111	7089	8198 683	7910 549	14999 0	14999 0	AbsAvg-MAB
111	8423	6576 705	8721 695	14999 0	9838 1430	AbsIns-SLMAB
112	182	8463 818	8593 753	8577 737	8645 862	NormAvg-SLMAB
112	7083	8369 891	7916 635	14999 0	14999 0	AbsIns-MAB
113	7052	9044 840	7947 1267	14999 0	14999 0	AbsExt-MAB
114	845	8347 596	8899 808	9192 862	8395 721	AbsAvg-SLMAB

Table 6.25: Results of each AOS scheme on the original OneMax function and on three other functions defined by monotonous transformations over it. The schemes were ranked on each function, and the first column presents the sum of their ranks, which defines the order of their presentation in the table. The second column depicts the difference between the highest and the lowest average performance of each AOS scheme over all the four functions.

monotonous transformations. Although the comparison-based counterparts FAUC-RMAB and FSR-RMAB are less competitive, their invariance property is verified (exactly the same performance on all the functions, $(h - l)$=0); this might show to be more beneficial in a bigger and more difficult class of problems. For all the other approaches, the best results in terms of robustness are achieved using Normalized versions of the Extreme (or

Instantaneous for DMAB) *Credit Assignment* schemes; this also confirms the expectations after discussion in Section 5.2.3.

Between the *Operator Selection* techniques other than RMAB, the baseline probability-based AP approach is the most robust: all its combinations are ranked in the first half of the Table. Its best combination in terms of performance, the NormExt-AP, achieves the best result in three out of four functions; however, it does not cope well with such a simple transformation as the logarithmic one. This result clearly demonstrates that the lack of invariance under simple nonlinear transformation could eventually cause some serious loss of efficiency for more difficult problems.

In what concerns the other bandit-based approaches using *Credit Assignment* schemes based on the raw values of fitness improvements, only the NormIns-DMAB is surprisingly able to perform well on this experimental scenario. A huge variance is shown in the results of the other combinations involving DMAB and MAB. Although presenting a smaller variance, SLMAB is still a bad choice, due to its non-competitive performance, populating mostly the bottom of the ranked Table, together with MAB.

Robustness on the BBOB Functions

An alternative analysis of the robustness of each AOS technique with respect to its hyper-parameters was done in the context of the BBOB benchmark scenario. Here, instead of using the same hyper-parameter configuration over different problems, the opposite approach is taken: different off-line tuning procedures were done, considering different groups of functions, and the best hyper-parameter configurations found for each of them are compared. A robust technique should present similar best hyper-parameter setting on most of the different cases, while still presenting competitive performance.

In the same way as for the empirical performance analysis presented in Section 6.6, the NDCG/FAUC-RMAB is used here as a representative of all the rank-based AOS schemes, being simply referred to as FAUC-B. MAB and SLMAB are disregarded here, due to their poor performance; the robustness of FAUC-B is compared to those of PM-AdapSS-DE, AP and DMAB, these two latter being coupled with the AbsExt *Credit Assignment* scheme.

Six different tuning procedures were performed for each dimension $d \in \{5, 20\}$, which are: tuning considering independently each of the 5 function classes; and tuning considering all functions. The best configuration found for each technique on each of the analyzed cases is presented in Table 6.26.

This tuning experiment confirms again the higher robustness of FAUC-B: $\{C = .5, W = 50\}$ is always the best configuration; except for the *multi-modal* and *weak-structure* class functions, in which none of the techniques was able to perform well, as discussed in Section 6.6. For the PM-AdapSS-DE, the benefits of using a relative instead of a raw reward are also shown on dimension 20, with always a very low value for p_{min}, and a high one for the adaptation rate α.

For AP, however, several configurations reached the end of the Racing process, all of them sharing $P.2$ and $W500$, but presenting all possible combinations for the adaptation rate α and the learning rate β. This could be seen as a good sign, possibly showing the robustness of this AOS combination; however, the use of p_{min}=0.2 in fact confirms that the

$d = 5$	FAUC-B	PM-AdapSS-DE	AbsExt-AP	AbsExt-DMAB
separable	C.5W50	P.05α.9	P.2α.6β.6W500	C10γ.01W100
moderate	C.5W50	P.05α.3	P.2α.9β.3W50	C.01G100W50
ill-conditioned	C.5W50	P0α.9	P.2α.6β.3W500	C100G1000W500
multi-modal	C.5W50	P.05α.9	P.2α.9β.3W500	C100G1W50
weak-structure	C1W500	P.05α.1	P.2α.9β.6W50	C100γ.1W50
all functions	C.5W50	P0α.9	P.2α.3β.6W500	C100γ.1W50
$d = 20$	FAUC-B	PM-AdapSS-DE	AbsExt-AP	AbsExt-DMAB
separable	C.5W50	P0α.9	P.2α.3β.1W500	C100γ.01W50
moderate	C.5W50	P0α.9	P.2α.9β.3W500	C100γ.01W50
ill-conditioned	C.5W50	P0α.9	P.2α.6β.3W500	C100γ.01W50
multi-modal	C1W50	P0α.3	P.2α.3β.1W100	C100γ.01W50
weak-structure	C.01W50	P0α.9	P.2α.3β.3W100	C100γ.01W50
all functions	C.5W50	P0α.6	P.2α.1β.1W500	C100γ.1W50

Table 6.26: Robustness analysis: best hyper-parameters configuration found for each technique on the BBOB benchmark set for dimensions 5 and 20, off-line tuned under different conditions.

method is presenting a behavior very close to the Naive uniform approach: as 4 mutation strategies are considered in this experimental scenario, the completely uniform behavior would be achieved with p_{min}=0.25, regardless the other parameters.

The same kind of conclusions can be drawn for the tuning experiments of DMAB. The configurations found for the different situations were all quite similar, with $C100$, $\gamma \leq .1$ and $W50$. However, a very high scaling factor C was found to be the best; this means that much more weight was given to the exploration term of the UCB formula, *i.e.*, although knowing which is the current best strategy, the algorithm prefers to explore the others. Besides, a very low value for the Page-Hinkley change detection threshold γ was chosen in most cases; this means that the probability of having restarts during the search is really high, also favoring the exploration, consequently dramatically degrading the performance of the method.

After all, the single fact that the same hyper-parameter tuning is found to be the best on different situations is not sufficient to conclude that a given technique is robust. Intuitively, if the final performance is as good as the uniform one, the configurations found are meaningless. The FAUC-B and the PM-AdapSS-DE, while presenting similar configurations for different situations, also perform very well, as shown in the empirical comparison presented in Section 6.6.

6.8 General Discussion

The use of so many and distinct benchmark scenarios was motivated by the possibility of analyzing different properties of the AOS schemes. On the artificial scenarios (Section 6.3), for instance, the agility to adapt to completely different situations was assessed under different conditions with respect to the definition of the artificial reward distributions and

to the level of informativeness of the received rewards. On the other hand, the results on the boolean benchmark problems (Section 6.4) represent the preliminary experiments in which the AOS schemes were analyzed *in situ*, selecting between different evolutionary operators within a real EA, applied to fitness landscapes with very different characteristics and levels of complexity. In Section 6.5, the DMAB *Operator Selection* technique was evaluated in the light of the hard combinatorial Boolean Satisfiability (SAT) problems, by using a third party *Credit Assignment* scheme that aggregates both fitness and diversity. Finally, on the Black-Box Optimization Benchmarking (BBOB) scenario, not only the performance of the techniques was assessed, but also their robustness, given the very heterogeneous benchmark function set provided by it; besides, in this case, yet another problem domain was tackled, the continuous one, by coupling the AOS schemes with a Differential Evolution (DE) algorithm.

After the description of the contributions for AOS in Chapter 5, these experiments provided enough empirical evidence to draw, under different benchmarking conditions, the following conclusions, somehow matching most of our expectations:

1. *DMAB Operator Selection technique performs better than MAB, AP, Naive uniform, and possibly equivalently to Oracle whenever available:*
   True. Indeed, DMAB is the overall winner technique in most cases. The price to pay for this gain in performance with respect to the standard MAB is the need to tune two very sensitive and problem-dependent hyper-parameters, the scaling factor C and the change-detection threshold γ. Given this mentioned problem-dependency, DMAB consequently does not perform well on scenarios considering many different problems, such as BBOB.

2. *SLMAB Operator Selection technique performs equivalently or better than DMAB, while having one hyper-parameter less:*
   False. This is the only notable deception with respect to the original expectation. In fact, SLMAB performed equivalently or better than DMAB only in some artificial scenarios (Section 6.3), and on the experiments within BBOB, in which both of DMAB and SLMAB performed rather poorly. In the other cases using real EAs, however, SLMAB was not able to efficiently follow the dynamics of the operator qualities (see, *e.g.*, its behavior plot on the OneMax problem, in Figure 6.12d), being outperformed even by the standard MAB in some cases. Besides, as for the other bandit-based approaches, there is still the need to tune the (also very sensitive) scaling factor C.

3. *Extreme Credit Assignment scheme performs better than the Instantaneous and Average ones:*
   True. Except for the artificial scenarios with small ΔT, in which more up-to-date information is needed in order to follow the very quick abrupt changed (thus preferring the Instantaneous scheme), Extreme performs better in most cases, including the boolean benchmark and the SAT problems. On the BBOB scenario, however, only Extreme was tried to date; the evaluation of the Average and Instantaneous counterparts is left for further work.

4. *Normalized versions perform equivalently or better than the Absolute versions of the Credit Assignment schemes, while being more robust with respect to its hyper-parameters:*
   True. In terms of performance, on the analyzed scenarios, they performed equivalently in most cases, while being better in a few cases. The gain in robustness by the use of this simple normalization scheme is clearly shown in the analysis on the transformed OneMax functions, presented in Section 6.7.2. However, it can be said to be significantly outperformed, in terms of robustness, by the rank-based schemes, which should thus be preferred, as discussed in the following.

5. *RMAB Operator Selection with the rank-based Credit Assignment schemes perform equivalently or better than the other AOS combinations, while being very robust with respect to their hyper-parameters:*
   True. Even in the artificial and boolean benchmark problems, in which the methods were off-line tuned for each problem (thus hindering the effects of the higher robustness), the different combinations of RMAB showed to be able of following closely the performance of the other AOS combinations in most cases. And, intuitively, the more heterogeneous the scenario, *i.e.*, the more different problems need to be tackled by the same hyper-parameter configuration, the clearer the benefits brought by the higher robustness, as shown in the BBOB scenario, in which RMAB is the clear winner. It is important to note that, besides the robustness, most of the efficiency in adaptation is also due to the *Credit Assignment* schemes in this case, while the bandit-based *Operator Selection* technique is responsible for controlling the EvE balance with respect to the operator selection. Furthermore, between the AUC and SR, the former showed to outperform the latter in the vast majority of the cases, being equivalent otherwise; then, between the AUC and the FAUC, the latter should be preferred just in case the robustness with respect to monotonous transformations is needed, otherwise the AUC should be employed.

Summarizing all these empirical evidences, to date, the AUC-RMAB remains as the recommended technique in case one wants to employ the AOS paradigm on his own algorithm. The last choice that needs to be made, between the Decay and the NDCG alternatives, is not critical and depends on whether there is some available budget for the tuning of the decay factor D or not, as the NDCG has shown to present similar but slightly inferior performance in most cases.

Part IV

General Conclusion

Chapter 7

Final Considerations

Evolutionary Algorithms (EAs) are stochastic algorithms that tackle search and optimization problems based on the Darwinian evolution paradigm. EAs have already shown to perform well in many different domains of application that are not tractable by standard methods, mainly because they do not make any strong assumption about the problem to be solved, and also due to the many parameters that enable the user to adapt the algorithm to the problem at hand, as discussed in Chapter 2. These many parameters, although providing the mentioned flexibility, are the main responsible factor for the fact that EAs are rarely used by researchers from other domains, as there are no general guidelines for their setting.

After the survey on parameter setting in EAs presented in Chapter 3, instead of using an expensive off-line tuning technique, the behavior of the algorithm should be rather adapted while solving the problem, according to the current needs of the search with respect to the Exploration versus Exploitation (EvE) balance. Taking as an example the choice of which operator should be applied, perturbative operators should be used to explore the search space in the initial stages of the search or when there is the need of escaping from stagnation, while fine-tuning operators should be preferred whenever there are promising regions that need to be further verified.

The use of feedback from the search to on-line adapt the selection of the operator to be applied is commonly referred to as Adaptive Operator Selection (AOS). In this work, we proposed different contributions to AOS, which will be summarized in Section 7.1. Section 7.2 will conclude this manuscript by sketching possible directions for further work.

7.1 Summary of Contributions

In order to perform AOS, one needs to define two elements, as described in Chapter 4 and depicted in Figure 4.1. After an operator application, the *Credit Assignment* scheme transforms its impact on the search process into a numerical reward, which is used to maintain an up-to-date estimation of its performance. Based on these empirical estimates, the *Operator Selection* mechanism selects the next operator to be applied.

Different approaches for both *Credit Assignment* and *Operator Selection* components

have been proposed in this thesis, resulting in novel AOS methods. These contributions are detailed in Chapter 5, and can be briefly summarized as follows.

The proposed *Credit Assignment* schemes use as a measure of impact the fitness improvement of the offspring with respect to its parent (or to the best of its parents in the case of crossover operators). The first proposal, simply referred to as Extreme *Credit Assignment*, rewards the operator with the maximum fitness improvement recently achieved by it, based on the assumption that outlier high improvements might be equally or more important than frequent but moderate ones. Although showing to be very efficient, this scheme provides a very problem-dependent behavior to the AOS methods for two reasons: (i) different problems have different fitness ranges, consequently reward ranges; and (ii) the magnitude of the fitness improvements tends to vary as the search advances (the closer to the optimum, usually the rarer and smaller the improvements). In the quest for a higher robustness, a simple normalization scheme was first proposed. Although alleviating the problem-dependency, the Normalized schemes are still based on the raw values of fitness improvements to some extent. This led us to further propose two *Credit Assignment* schemes based on ranks over the fitness improvements, which showed to be very robust: the first method is inspired by a Machine Learning paradigm, referred to as Area-Under-Curve (AUC), while the second simply uses the Sum-of-Ranks (SR) to evaluate the operator qualities. An extra level of robustness (with a small price to pay in terms of efficiency) was further achieved by the use of ranks over fitness values instead of ranks over fitness improvements: the resulting algorithms are totally comparison-based, *i.e.*, invariant with respect to monotonous transformations over the original fitness function. The respective methods are referred to as Fitness-based Area-Under-Curve (FAUC) and Fitness-based Sum-of-Ranks (FSR).

The *Operator Selection* issue was tackled as another instance of the Exploration versus Exploitation (EvE) dilemma: the operator that is found to perform better than the others should be used as much as possible (exploitation), while other operators should also be tried from time to time (exploration), as one of them might become the new best one at a further (unknown) instant of the search. This dilemma has been intensively studied in the context of yet another Machine Learning paradigm (more specifically in Game Theory), the Multi-Armed Bandit (MAB). A first tentative application of this paradigm to the *Operator Selection* problem used the Upper Confidence Bound (UCB) algorithm [Auer *et al.*, 2002], which provides asymptotic optimality guarantees with respect to total cumulated reward; but these guarantees hold only in the original and stationary context of MAB problems, while the AOS context is very dynamic: the performance of the operators continuously vary as the search goes on. As a consequence, different proposals were made in order to be able to efficiently use the MAB paradigm in the AOS context. The first proposal on this direction, referred to as Dynamic Multi-Armed Bandit (DMAB), uses a statistical test to trigger a restart of the MAB process whenever a change on the operator quality distribution is detected. This method has shown to be very efficient, but different problems have fitness landscapes with different dynamics, what makes its restarting mechanism to be also highly problem-dependent. This led us to propose the Sliding Multi-Armed Bandit (SLMAB), which accounts for the AOS dynamics by continuously adapting the weight of the received rewards, according to how frequent each operator has been applied: the less frequent,

the more outdated is its corresponding empirical estimate, consequently the higher should be the weight of the instant reward received, and vice-versa. In practice, however, this mechanism did not show to perform as good as expected. The last proposal, referred to as RMAB, is indeed the simplest one; the AOS dynamics are in fact handled on the *Credit Assignment* side by the rank-based schemes in a transparent way: as the ranking considers the rewards received by all operators, the application of one operator affects the perceived quality of all the others; in this way, their quality estimates are always sufficiently up-to-date by construction.

A last contribution concerns the empirical assessment of the resulting AOS methods. Firstly, some artificial scenarios were proposed to analyze their behavior on different controlled environments. And secondly, a very extensive empirical analysis of all the proposed AOS methods, also compared with some baseline methods, was performed and analyzed in detail (Chapter 6): they were assessed within a Genetic Algorithm applied to the proposed artificial scenarios, to other boolean benchmark problems, as well as to the hard Boolean Satisfiability problems; and within a Differential Evolution algorithm applied to a comprehensive benchmark set of continuous functions.

Based on the empirical evidences gathered from this analysis, in case one wants to apply the AOS paradigm to his own algorithm/problem, the recommended AOS method as of today is the combination of the AUC *Credit Assignment* scheme with the RMAB *Operator Selection* mechanism: it achieves state-of-the-art performance, while also being very robust with respect to its hyper-parameters when applied to different problems.

7.2 Further Work

A major drawback of the final recommended AOS method is that its remarkable robustness and its state-of-the-art performance remains limited, mostly, to unimodal problems. In order to efficiently tackle multi-modal problems, the impact of the operator application with relation to the population diversity should also be considered somehow, while just the fitness is being currently regarded. This explains why none of the methods were able to perform well on the multi-modal functions of the BBOB testbed. Some preliminary work has been done on this direction (Section 6.5), using the Compass method from Universitè d'Angers as *Credit Assignment*, assessed within a GA in the context of Boolean Satisfiability (SAT) problems (see Section 6.5). But in this work, only the very sensitive and problem-dependent DMAB *Operator Selection* technique was used. Further work should concern the preservation of the achieved level of robustness and efficiency in the framework of the Pareto Dominance-based *Credit Assignment* scheme proposed in [Maturana *et al.*, 2010b], which is a follow-up of the *Compass* method.

An alternative approach for tackling multi-modal problems can be found in the state-of-the-art optimizer IPOP-CMA-ES [Auger and Hansen, 2005], which was used as a baseline for comparison on the experiments within BBOB. Instead of modifying its adaptive mechanism, it uses a deterministic parameter control of the population size: after some number of generations, if no improvement has been achieved, the size of the population is doubled and the search is restarted from scratch. Bigger the population, higher are the chances

of finding the global optimum, as it enables a better parallel exploration of the multiple peaks of the fitness landscape. This approach should be tried in the near future, by incorporating the restarting/population size control mechanism into the underlying algorithms that were tried with the AOS schemes, namely, GAs and DE.

Coming back to the work on SAT problems, the AOS methods were used to select between rather naive operators, thus not achieving competitive performance. In order to possibly take part in SAT races, further work should concern, thus, the autonomous selection between state-of-the-art operators for SAT. Indeed, we are currently working in collaboration with University of British Columbia on this topic; but, as of today, there are no conclusive results yet.

Along the same line, in the work within the DE algorithm applied to continuous problems, the AOS methods were combined with the very standard version of DE. Several more efficient DE variants exist nowadays, such as the JADE algorithm [Zhang and Sanderson, 2009], which, besides using improved mutation strategies, also employs the on-line adaptation of some of the DE parameters, namely, the mutation scaling factor F and the crossover rate CR. As a continuation of the collaboration work with the China University of Geosciences, the PM-AdapSS-DE method (Section 6.6.3) was tried within JADE, achieving significantly better results than when combined with the standard algorithm ([Gong *et al.*, 2010b], currently under review). A natural next step in this case would be to try our rank-based AOS methods with JADE, in order to possibly achieve more competitive results when compared to state-of-the-art optimizers such as the IPOP-CMA-ES [Auger and Hansen, 2005].

Another path for further work, that is also being currently explored in the scope of a collaboration, this time with the City University of Hong Kong, involves the DE algorithm again, but applied to multi-objective problems. This work is still in a very preliminary stage, and the main difficulty for the time being is, as in the standard non-adaptive framework, to define how to efficiently evaluate the quality of the solutions (and consequently the impact of the operator application) with respect to the different objectives.

As it can be seen from all these on-going collaborations, the AOS paradigm is indeed very useful and general enough to be applied to many different contexts. But up to now, we have considered their application only to the selection of operators within EAs. The developed AOS methods should also be further assessed within different kinds of metaheuristics and stochastic algorithms; and at a higher level of abstraction, selecting between heuristics instead of operators, what is commonly referred to as Hyper-Heuristics. For instance, the upcoming "International Cross-Domain Heuristic Search Challenge"[1] seems to be an interesting experimental framework to evaluate the AOS mechanisms at the level of Hyper-Heuristics. On this challenge, optimization algorithms will be evaluated over several kinds of combinatorial problems (MAX-SAT, bin-packing, flow-shop and scheduling); a promising approach would be to feed the AOS mechanism with heuristics that perform well on each of these domains, and then let it on-line select between them, hopefully discovering and exploiting the heuristics that are more adapted for each problem domain.

[1] `http://www.asap.cs.nott.ac.uk/chesc2011`

Bibliography

[Abdullah *et al.*, 2007] cited page(s) 25
S. Abdullah, E.K. Burke, and B. McCollum. A hybrid evolutionary approach to the university course timetabling problem. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 1764–1768, 2007.

[Angeline, 1995] cited page(s) 41
P.J. Angeline. Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence: A Dynamic Systems Perspective*, pages 152–161. IEEE, 1995.

[Angelov *et al.*, 2003] cited page(s) 26
P.P. Angelov, Y. Zhang, J.A. Wright, V.I. Hanby, and R.A. Buswell. Automatic design synthesis and optimization of component-based systems by evolutionary algorithms. In E. Cantú-Paz et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, volume 2724 of *LNCS*, pages 1938–1950. Springer, 2003.

[Arabas *et al.*, 1994] cited page(s) 32
J. Arabas, Z. Michalewitz, and J. Mulawka. GAVAPS - a genetic algorithm with varying population size. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 73–78. IEEE, 1994.

[Arbib, 2002] cited page(s) 12
M.A. Arbib, editor. *The Handbook of Brain Theory and Neural Networks*. MIT, 2002.

[Auer *et al.*, 2002] cited page(s) 5, 52, 68, 68, 79, 80, 82, 186
P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[Auger and Hansen, 2005] cited page(s) 94, 160, 166, 187, 188
A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In B. McKay et al., editor, *Proc. IEEE Congress on Evolutionary Computation (CEC)*, volume 2, pages 1769–1776. IEEE, 2005.

[Auger and Teytaud, 2010] cited page(s) 30
A. Auger and O. Teytaud. Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica*, 57(1):121–146, 2010.

[Bäck *et al.*, 2000] cited page(s) 32
T. Bäck, A.E. Eiben, and N.A.L. van der Vaart. An empirical study on GAs without parameters. In M. Schoenauer et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*, volume 1917 of *LNCS*, pages 315–324. Springer, 2000.

[Bäck, 1992] cited page(s) 39
T. Bäck. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In R. Männer et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*, pages 87–96. Elsevier, 1992.

[Balaprakash *et al.*, 2007] cited page(s) 36, 36, 38, 103
P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In T. Bartz-Beielstein et al., editor, *Hybrid Metaheuristics*, volume 4771 of *LNCS*, pages 108–122. Springer, 2007.

[Barbosa and Sá, 2000] cited page(s) 48, 49, 50, 52, 57
H.J.C. Barbosa and A.M. Sá. On adaptive operator probabilities in real coded genetic algorithms. In *Workshop on Advances and Trends in Artificial Intelligence for Problem Solving, XX Intl. Conf. of the Chilean Computer Science Society (SCCC)*, 2000.

[Bartz-Beielstein *et al.*, 2005] cited page(s) 37, 103
T. Bartz-Beielstein, C.W.G. Lasarczyk, and M. Preuss. Sequential parameter optimization. In B. McKay et al., editor, *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 773–780. IEEE, 2005.

[Battiti *et al.*, 2008] cited page(s) 40
R. Battiti, M. Brunato, and F. Mascia. *Reactive Search and Intelligent Optimization*. Operations research/Computer Science Interfaces. Springer, 2008.

[Beyer, 1995] cited page(s) 41
H.-G. Beyer. Toward a theory of evolution strategies: Self-adaptation. *Evolutionary Computation*, 3(3):311–347, 1995.

[Bibai *et al.*, 2010] cited page(s) 25
J. Bibai, P. Savéant, M. Schoenauer, and V. Vidal. An evolutionary metaheuristic based on state decomposition for domain-independent satisficing planning. In R. Brafman et al., editor, *Proc. Intl. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 15–25. AAAI Press, 2010.

[Birattari *et al.*, 2002] cited page(s) 36, 36, 36, 36, 36, 102, 102
M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W.B. Langdon et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 11–18. Morgan Kaufmann, 2002.

[Birattari *et al.*, 2009] cited page(s) 38
M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. Automated algorithm tuning using F-Races: Recent developments. In M. Caserta et al., editor, *Proc. Metaheuristic Intl. Conference*. University of Hamburg, 2009.

[Birattari, 2004a] cited page(s) 102
M. Birattari. On the estimation of the expected performance of a metaheuristic on a class of instances. How many instances, how many runs? Technical Report TR/IRIDIA/2004-01, IRIDIA, Université Libre de Bruxelles, Belgium, Brussels, Belgium, 2004.

[Birattari, 2004b] cited page(s) 102
M. Birattari. *The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective*. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium, 2004.

[Bradley, 1997] cited page(s) 75
A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997.

[Branke *et al.*, 2003] cited page(s) 18
J. Branke, C. Barz, and I. Behrens. Ant-based crossover for permutation problems. In E. Cantú-Paz et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 754–765. Springer, 2003.

[Burges *et al.*, 2005] cited page(s) 74
C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proc. Intl. Conf. on Machine Learning (ICML)*, pages 89–96. ACM, 2005.

[Burke *et al.*, 2010] cited page(s) 41, 62
E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu. Hyper-heuristics: A survey of the state of the art. Technical Report NOTTCS-TR-SUB-0906241418-2747, School of Computer Science and Information Technology, University of Nottingham, 2010.

[Chan *et al.*, 2005] cited page(s) 18
C.-H. Chan, S.-A. Lee, C.-Y. Kao, and H.-K. Tsai. Improving EAX with restricted 2-opt. In H.-G. Beyer et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 1471–1476. ACM, 2005.

[Christensen and Oppacher, 2001] cited page(s) 30
S. Christensen and F. Oppacher. What can we learn from No Free Lunch? A first attempt to characterize the concept of a searchable function. In L. Spector et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 1219–1234. Morgan Kaufmann, 2001.

[Clune *et al.*, 2005] cited page(s) 37
J. Clune, S. Goings, B. Punch, and E. Goodman. Investigations in Meta-GA: panaceas or pipe dreams? In H.-G. Beyer et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 235–241. ACM, 2005.

[Collet and Schoenauer, 2003] cited page(s) 3
P. Collet and M. Schoenauer. GUIDE: Unifying evolutionary engines through a graphical user interface. In P. Liardet et al., editor, *Proc. Intl. Conf. on Artificial Evolution (EA)*, volume 2936 of *LNCS*, pages 203–215. Springer, 2003.

[Conover, 1999] cited page(s) 102
W.J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, 1999.

[Cook, 1971] cited page(s) 15, 151
S.A. Cook. The complexity of theorem-proving procedures. In *Proc. ACM Symposium on Theoryof Computing (STOC)*, pages 151–158. ACM, 1971.

[Da Costa and Schoenauer, 2009] cited page(s) 3, 20
L. Da Costa and M. Schoenauer. Bringing evolutionary computation to industrial applications with guide. In F. Rothlauf et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 1467–1474. ACM, 2009.

[Da Costa *et al.*, 2008] cited page(s) 5, 50, 68, 69, 80, 84, 84, 90, 105, 170
L. Da Costa, Á. Fialho, M. Schoenauer, and M. Sebag. Adaptive operator selection with dynamic multi-armed bandits. In C. Ryan et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 913–920. ACM, 2008.

[Das *et al.*, 2009] cited page(s) 25
S. Das, A. Abraham, U.K. Chakraborty, and A. Konar. Differential evolution using a neighborhood-based mutation operator. *IEEE Transactions on Evolutionary Computation*, 13(3):526–553, 2009.

[Davis, 1989] cited page(s) 48, 48, 49, 50, 52, 56, 56, 56
L. Davis. Adapting operator probabilities in genetic algorithms. In J.D. Schaffer et al., editor, *Proc. Intl. Conf. on Genetic Algorithms (ICGA)*, pages 61–69. Morgan Kaufmann, 1989.

[Deb, 2001] cited page(s) 15, 26
K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, 2001.

[DeJong and Spears, 1990] cited page(s) 30, 30, 30, 30
K. DeJong and W.M. Spears. An analysis of the interacting roles of population size and crossover in genetic algorithms. In H.-P. Schwefel et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*, pages 38–47. Springer, 1990.

[DeJong, 2006] cited page(s) 3, 12, 18, 21, 31
K. DeJong. *Evolutionary Computation. A unified Approach*. MIT, 2006.

[DeJong, 2007] cited page(s) 19, 20, 31, 32, 33, 33, 33, 34, 39, 42
K. DeJong. Parameter setting in EAs: a 30 year perspective. In Lobo et al. [2007], pages 1–18.

[Dorigo *et al.*, 1996] cited page(s) 12, 12
M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, 26(1):29–41, 1996.

[Eberhart *et al.*, 2001] cited page(s) 12, 12
R.C. Eberhart, Y. Shi, and J. Kennedy. *Swarm Intelligence*. Morgan Kaufmann, 1st edition, 2001.

[Eiben and Schoenauer, 2002] cited page(s) 14, 14, 16, 16
A.E. Eiben and M. Schoenauer. Evolutionary computing. *Information Processing Letters*, 82(1):1–6, 2002.

[Eiben and Smith, 2003] cited page(s) 3, 16, 28
A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[Eiben and van Hemert, 1999] cited page(s) 41
A.E. Eiben and J.I. van Hemert. SAW-ing EAs: adapting the fitness function for solving constrained problems. In D. Corne et al., editor, *New ideas in optimization*, chapter 26, pages 389–402. McGraw-Hill, 1999.

[Eiben *et al.*, 1999] cited page(s) 4, 31, 34, 35, 35
A.E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[Eiben *et al.*, 2004] cited page(s) 32
A.E. Eiben, E. Marchiori, and V. Valko. Evolutionary algorithms with on-the-fly population size adjustment. In X. Yao et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*. Springer, 2004.

[Eiben *et al.*, 2006] cited page(s) 32, 33, 33
A.E. Eiben, M.C. Schut, and A.R. de Wilde. Is self-adaptation of selection pressure and population size possible? In T.P. Runarsson et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*, volume 4193 of *LNCS*, pages 900–909. Springer, 2006.

[Eiben *et al.*, 2007] cited page(s) 4, 4, 4, 4, 30, 34, 39, 39, 41, 46
A.E. Eiben, Z. Michalewicz, M. Schoenauer, and J.E. Smith. Parameter control in evolutionary algorithms. In Lobo et al. [2007], pages 19–46.

[Eiben, 2002] cited page(s) 27
A.E. Eiben. Evolutionary computing: the most powerful problem solver in the universe? *Dutch Mathematical Archive*, 5/3:126–131, 2002.

[Fialho and Ros, 2010] cited page(s) 62, 77, 89, 158, 160, 170
Á. Fialho and R. Ros. Analysis of adaptive strategy selection within differential evolution on the BBOB-2010 noiseless benchmark. Research Report RR-7259, INRIA, 2010.

[Fialho *et al.*, 2008] cited page(s) 50, 68, 69, 71, 84, 133, 134
Á. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Extreme value based adaptive operator selection. In G. Rudolph et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*, volume 5199 of *LNCS*, pages 175–184. Springer, 2008.

[Fialho *et al.*, 2009a] cited page(s) 69, 84, 133, 141
Á. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In T. Stützle et al. [2009], pages 176–190.

[Fialho *et al.*, 2009b] cited page(s) 50, 69, 72, 84, 141
Á. Fialho, M. Schoenauer, and M. Sebag. Analysis of adaptive operator selection techniques on the royal road and long k-path problems. In F. Rothlauf et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 779–786. ACM, 2009.

[Fialho *et al.*, 2010a] cited page(s) 5, 69, 69, 85, 87, 87, 90, 90, 91, 92, 105, 170
Á. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence – Special Issue on Learning and Intelligent Optimization*, 2010.

[Fialho *et al.*, 2010b] cited page(s) 6, 62, 69, 73, 77, 89, 158, 158, 160
Á. Fialho, R. Ros, M. Schoenauer, and M. Sebag. Comparison-based adaptive strategy selection in differential evolution. In R. Schaefer et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*. Springer, 2010.

[Fialho *et al.*, 2010c] cited page(s) 6, 69, 69, 73, 77, 89, 105, 133, 170
Á. Fialho, M. Schoenauer, and M. Sebag. Toward comparison-based adaptive operator selection. In M. Pelikan et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2010.

[Fogel *et al.*, 1966] cited page(s) 12, 22
L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, 1966.

[Fogel, 1995] cited page(s) 12, 22
D.B. Fogel. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. IEEE, 1995.

[Fogel, 1998] cited page(s) 12
D.B. Fogel. *Evolutionary Computing: The Fossile Record*. IEEE, 1998.

[Gagliolo and Schmidhuber, 2008] cited page(s) 68
M. Gagliolo and J. Schmidhuber. Algorithm selection as a bandit problem with unbounded losses. Technical Report IDSIA - 07 - 08, IDSIA, 2008.

[Garnier and Kallel, 2000] cited page(s) 140, 140, 141, 142
J. Garnier and L. Kallel. Statistical distribution of the convergence time of evolutionary algorithms for long-path problems. *IEEE Transactions on Evolutionary Computation*, 4(1), 2000.

[Gelly *et al.*, 2007] cited page(s) 6
S. Gelly, S. Ruette, and O. Teytaud. Comparison-based algorithms are robust and randomized algorithms are anytime. *Evolutionary Computation*, 15(4):411–434, 2007.

[Giger *et al.*, 2007] cited page(s) 48, 49, 49, 59
M. Giger, D. Keller, and P. Ermanni. AORCEA - an adaptive operator rate controlled evolutionary algorithms. *Computers & Structures*, 85(19-20):1547 – 1561, 2007.

[Goldberg *et al.*, 1991] cited page(s) 34
D.E. Goldberg, K. Deb, and B. Korb. Don't worry, be messy. In R. K. Belew et al., editor, *Proc. Intl. Conf. on Genetic Algorithms (ICGA)*, pages 24–30. Morgan Kaufmann, 1991.

[Goldberg *et al.*, 1992] cited page(s) 32
D.E. Goldberg, K. Deb, and J.H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992.

[Goldberg *et al.*, 1995] cited page(s) 20
D.E. Goldberg, H. Kargupta, J. Horn, and E. Cantu-Paz. Critical deme size for serial and parallel genetic algorithms. Technical Report IlliGAL Report 95002, IlliGAL, 1995.

[Goldberg, 1989] cited page(s) 12, 22
D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, 1989.

[Goldberg, 1990] cited page(s) 52, 52
D.E. Goldberg. Probability matching, the magnitude of reinforcement, and classifier system bidding. *Machine Learning*, 5(4):407–426, 1990.

[Gong *et al.*, 2010a] cited page(s) 62, 158, 159, 160, 169
W. Gong, Á. Fialho, and Z. Cai. Adaptive strategy selection in differential evolution. In M. Pelikan et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2010.

[Gong *et al.*, 2010b] cited page(s) 62, 188
W. Gong, Á. Fialho, Z. Cai, and H. Li. Adaptive strategy selection in differential evolution for numerical optimization. *Information Sciences*, 2010. (currently under review).

[Grefenstette, 1986] cited page(s) 30, 30, 30, 30
J.J. Grefenstette. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, 16:122–128, 1986.

[Hansen and Ostermeier, 2001] cited page(s) 6, 22, 25, 33, 40, 41
N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[Hansen *et al.*, 2009a] cited page(s) 159, 160, 163
N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009. Updated Feb 2010.

[Hansen *et al.*, 2009b] cited page(s) 159
N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions. Technical Report RR-6869, INRIA, 2009.

[Hansen *et al.*, 2009c] cited page(s) 25
N. Hansen, A. Niederberger, L. Guzzella, and P. Koumoutsakos. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Transactions on Evolutionary Computation*, 13(1):180–197, 2009.

[Hansen *et al.*, 2010a] cited page(s) 158
N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical Report RR-7215, INRIA, 2010.

[Hansen *et al.*, 2010b] cited page(s) 158
N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In M. Pelikan et al., editor, *GECCO (Companion)*. ACM, 2010.

[Hansen, 2008] cited page(s) 34
N. Hansen. Adaptive encoding: How to render search coordinate system invariant. In G. Rudolph et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*, pages 205–214. Springer, 2008.

[Hansen, 2009a] cited page(s) 160
N. Hansen. Benchmarking a bi-population CMA-ES on the BBOB-2009 noiseless testbed. In F. Rothlauf et al., editor, *GECCO (Companion)*, pages 2389–2396, 2009.

[Hansen, 2009b] cited page(s) 25
N. Hansen. References to CMA-ES applications, December 2009. Available at: http://www.lri.fr/ hansen/cmaapplications.pdf, Lastly accessed on: Oct 18th 2010.

[Hartland *et al.*, 2007] cited page(s) 84
C. Hartland, N. Baskiotis, S. Gelly, O. Teytaud, and M. Sebag. Change point detection and meta-bandits for online learning in dynamic environments. In *Proc. Conférence Francophone sur l'Apprentissage Automatique (CAPS)*, 2007.

[Hatta *et al.*, 1997] cited page(s) 57
K. Hatta, K. Matsuda, S. Wakabayashi, and T. Koide. On-the-fly crossover adaptation of genetic algorithms. In *Proc. Intl. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 197–202, 1997.

[Hatta *et al.*, 2001] cited page(s) 57, 57
K. Hatta, S. Wakabayashi, and T. Koide. Adaptation of genetic operators and parameters of a genetic algorithm based on the elite degree of an individual. *Systems and Computers in Japan*, 32(1):29–37, 2001.

[Herrera and Lozano, 1998] cited page(s) 33
F. Herrera and M. Lozano. Fuzzy genetic algorithms: Issues and models. Technical Report DECSAI-98116, University of Granada, Dept. of Computer Science and A.I., 1998.

[Herrera and Lozano, 2001] cited page(s) 60
F. Herrera and M. Lozano. Adaptive genetic operators based on coevolution with fuzzy behaviors. *IEEE Transactions on Evolutionary Computation*, 5(2):149–165, 2001.

[Hesser and Männer, 1990] cited page(s) 39
J. Hesser and R. Männer. Towards an optimal mutation probability for genetic algorithms. In H.-P. Schwefel et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*, pages 23–32. Springer, 1990.

[Hinkley, 1970] cited page(s) 5, 68
D.V. Hinkley. Inference about the change point from cumulative sum-tests. *Biometrika*, 58(3):509–523, 1970.

[Ho *et al.*, 1999] cited page(s) 48, 49, 58, 60
C.W. Ho, K.H. Lee, and K.S. Leung. A genetic algorithm based on mutation and crossover with adaptive probabilities. In P.J. Angeline et al., editor, *Proc. IEEE Congress on Evolutionary Computation (CEC)*, volume 1, pages 768–775. IEEE, 1999.

[Holland, 1975] cited page(s) 12, 22
J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[Holland, 1993] cited page(s) 145, 146
J.H. Holland. Royal road functions. In *Internet Genetic Algorithms Digest 7:22.* MIT, 1993.

[Hoos and Stützle, 2000] cited page(s) 152
H.H. Hoos and T. Stützle. *SATLIB: An Online Resource for Research on SAT*, pages 283–292. IOS Press, 2000. Available at: www.satlib.org, Lastly accessed on Oct 18th 2010.

[Horn *et al.*, 1994] cited page(s) 137
J. Horn, D.E. Goldberg, and K. Deb. Long path problems. In Y. Davidor et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*, pages 149–158. Springer, 1994.

[Horn, 1997] cited page(s) 27
J. Horn. *The nature of niching: genetic algorithms and the evolution of optimal, cooperative populations*. PhD thesis, University of Illinois at Urbana-Champaign, 1997.

[Hutter and Hamadi, 2005] cited page(s) 43
F. Hutter and Y. Hamadi. Parameter adjustment based on performance prediction: Towards an instance-aware problem solver. Technical Report MSR-TR-2005-125, Microsoft Research, 2005.

[Hutter *et al.*, 2006] cited page(s) 4
F. Hutter, Y. Hamadi, H.H. Hoos, and K. Leyton-Brown. Performance prediction and automated tuning of randomized and parametric algorithms. In *Proc. Intl. Conf. on Principles and Practice of Constraint Programming (CP)*, number 4204 in LNCS, pages 213–228. Springer, 2006.

[Hutter *et al.*, 2009] cited page(s) 37, 37
F. Hutter, H.H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36(1):267–306, 2009.

[Ibanez *et al.*, 2009] cited page(s) 25
O. Ibanez, L. Ballerini, O. Cordón, S. Damas, and J. Santamaría. An experimental study on the applicability of evolutionary algorithms to craniofacial superimposition in forensic identification. *Information Sciences*, 179(23):3998–4028, 2009.

[Igel *et al.*, 2007] cited page(s) 41
C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

[J. Romero et al., 2007] cited page(s) 26
J. Romero et al., editor. *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. Natural Computing Series. Springer, 2007.

[Jansen *et al.*, 2005] cited page(s) 32
T. Jansen, K.A. DeJong, and I. Wegener. On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation*, 13(4):413–440, 2005.

[Järvelin and Kekäläinen, 2000] cited page(s) 74
K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proc. Intl. ACM Conf. on Research and Development in Information Retrieval*, pages 41–48. ACM, 2000.

[Jones, 1994] cited page(s) 145
T. Jones. A description of Holland's Royal Road. *Evolutionary Computation*, 2(4):409–415, 1994.

[Julstrom, 1995] cited page(s) 48, 48, 49, 50, 52, 56, 56
B. Julstrom. What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm. In L.J. Eshelman et al., editor, *Proc. Intl. Conf. on Genetic Algorithms (ICGA)*, pages 81–87. Morgan Kaufmann, 1995.

[Julstrom, 1997] cited page(s) 48, 48, 49, 50, 52, 56, 56, 57
B.A. Julstrom. Adaptive operator probabilities in a genetic algorithm that applies three operators. In *Proc. ACM Symposium on Applied Computing (SAC)*, pages 233–238. ACM, 1997.

[Kamalian *et al.*, 2005] cited page(s) 26
R. Kamalian, Y. Zhang, H. Takagi, and A.M. Agogino. Reduced human fatigue in interactive evolutionary computation for micromachine design. In *Intl. Conf. on Machine Learning and Cybernetics*, pages 5666–5671. IEEE, 2005.

[Keijzer *et al.*, 2002] cited page(s) 3, 20, 93
M. Keijzer, J.J. Merelo, G. Romero, and M. Schoenauer. Evolving Objects: a general purpose evolutionary computation library. In P. Collet et al., editor, *Proc. Intl. Conf. on Artificial Evolution (EA)*, volume 2310 of *LNCS*, pages 229–241. Springer, 2002. Available at: http://eodev.sourceforge.net/, Lastly accessed on Oct 18th 2010.

[Klir and Yuan, 1995] cited page(s) 12
G.J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall, 1st edition, 1995.

[Koza *et al.*, 2000] cited page(s) 26
J.R. Koza, M.A. Keane, J. Yu, F.H. Bennett, and W. Mydlowec. Automatic creation of human-competitive programs and controllers by means of genetic programming. *Genetic Programming and Evolvable Machines*, 1(1-2):121–164, 2000.

[Koza *et al.*, 2003] cited page(s) 16
J.R. Koza, M.A. Keane, and M.J. Streeter. What's AI done for me lately? Genetic programming's human-competitive results. *IEEE Intelligent Systems*, 18(3):25–31, 2003.

[Koza, 1992] cited page(s) 12, 22
J. R. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Evolution*. MIT, 1992.

[Koza, 1994] cited page(s) 12, 22
J.R. Koza. *Genetic programming II : automatic discovery of reusable programs*. Complex adaptive systems. MIT, 1994.

[Koza, 2010] cited page(s) 26
J.R. Koza. Annual hummies awards, 2010. Available at: http://www.genetic-programming.org/hc2005/main.html, Lastly accessed on Oct 18th 2010.

[Krasnogor, 2002] cited page(s) 18
N. Krasnogor. *Studies on the Theory and Design Space of Memetic Algorithms*. PhD thesis, University of the West of England, 2002. Supervisor: Dr. J.E. Smith.

[Lai and Robbins, 1985] cited page(s) 5, 68, 79, 80
T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[Lardeux *et al.*, 2006] cited page(s) 15, 18
F. Lardeux, F. Saubion, and J.-K. Hao. GASAT: a genetic local search algorithm for the satisfiability problem. *Evolutionary Computation*, 14(2):223–253, 2006.

[Lee and Takagi, 1993] cited page(s) 60
M.A. Lee and H. Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. In *Proc. Intl. Conf. on Genetic Algorithms (ICGA)*, pages 76–83. Morgan Kaufmann, 1993.

[Lobo and Goldberg, 1997] cited page(s) 48, 49, 52, 57
F.G. Lobo and D.E. Goldberg. Decision making in a hybrid genetic algorithm. In B. Porto et al., editor, *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 121–125. IEEE, 1997.

[Lobo *et al.*, 2007] cited page(s) 4, 31, 192, 193
F.G. Lobo, C.F. Lima, and Z. Michalewicz, editors. *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*. Springer, 2007.

[Luchian and Gheorghies, 2003] cited page(s) 48, 49, 52, 58, 58
H. Luchian and O. Gheorghies. Integrated-adaptive genetic algorithms. In W. Banzhaf et al., editor, *Proc. European Conf. on Advances in Artificial Life*, volume 2801 of *Lecture Notes in Computer Science*, pages 635–642. Springer, 2003.

[Luke and Panait, 2006] cited page(s) 22
S. Luke and L. Panait. A comparison of bloat control methods for genetic programming. *Evolutionary Computation*, 14(3):309–344, 2006.

[Martikainen and Ovaska, 2006] cited page(s) 15
J. Martikainen and S.J. Ovaska. Fitness function approximation by neural networks in the optimization of MGP-FIR filters. In *IEEE Mountain Workshop on Adaptive and Learning Systems*, pages 7–12, 2006.

[Maturana and Saubion, 2007a] cited page(s) 60
J. Maturana and F. Saubion. On the design of adaptive control strategies for evolutionary algorithms. In N. Monmarché et al., editor, *Proc. Intl. Conf. on Artificial Evolution (EA)*, volume 4926 of *LNCS*. Springer, 2007.

[Maturana and Saubion, 2007b] cited page(s) 60
J. Maturana and F. Saubion. Towards a generic control strategy for evolutionary algorithms: an adaptive fuzzy-learning approach. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2007.

[Maturana and Saubion, 2008a] cited page(s) 4, 48, 49, 49, 51, 51, 51, 52, 58, 58, 59, 63, 150, 151, 152
J. Maturana and F. Saubion. A compass to guide genetic algorithms. In G. Rudolph et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*, volume 5199 of *LNCS*, pages 256–265. Springer, 2008.

[Maturana and Saubion, 2008b] cited page(s) 60
J. Maturana and F. Saubion. From parameter control to search control: Parameter control abstraction in evolutionary algorithms. *Constraint Programming Letters - Special Issue on Autonomous Search*, 4:39–65, 2008.

[Maturana *et al.*, 2009a] cited page(s) 4, 7, 51, 63, 69, 84, 150
J. Maturana, Á. Fialho, F. Saubion, M. Schoenauer, and M. Sebag. Extreme compass and dynamic multi-armed bandits for adaptive operator selection. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 365–372. IEEE, 2009.

[Maturana *et al.*, 2009b] cited page(s) 59
J. Maturana, F. Lardeux, and F. Saubion. Controlling behavioral and structural parameters in evolutionary algorithms. In *Proc. EA'09*, 2009.

[Maturana *et al.*, 2010a] cited page(s) 7, 51, 150
J. Maturana, Á. Fialho, F. Saubion, M. Schoenauer, F. Lardeux, and M. Sebag. Adaptive operator selection and management in evolutionary algorithms. In Y. Hamadi et al, editor, *Autonomous Search*. Springer, 2010. (to appear).

[Maturana *et al.*, 2010b] cited page(s) 49, 49, 59, 94, 156, 187
J. Maturana, F. Lardeux, and F. Saubion. Autonomous operator management for evolutionary algorithms. *Journal of Heuristics*, 2010.

[Maturana, 2009] cited page(s) 41
J. Maturana. *Generic Parameter Control for Evolutionary Algorithms*. PhD thesis, Université d'Angers, Angers, France, 2009.

[Merz and Freisleben, 1997] cited page(s) 16, 25
P. Merz and B. Freisleben. Genetic local search for the TSP: new results. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 159 –164, 1997.

[Michalewicz, 1996] cited page(s) 3
Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 3rd edition, 1996.

[Mitchell *et al.*, 1992] cited page(s) 145
M. Mitchell, S. Forrest, and J.H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In *Proc. European Conf. on Artificial Life (ECAL)*, pages 245–254, 1992.

[Mitchell, 1998] cited page(s) 23
M. Mitchell. *An Introduction to Genetic Algorithms*. MIT, 1998.

[Moore, 1991] cited page(s) 31
G.A. Moore. *Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customer*. Collins Business Essentials, 1991.

[Mueller-Gritschneder *et al.*, 2009] cited page(s) 15
D. Mueller-Gritschneder, H. Graeb, and U. Schlichtmann. A successive approach to compute the bounded pareto front of practical multiobjective optimization problems. *SIAM Journal on Optimization*, 20(2):915–934, 2009.

[Nannen and Eiben, 2007] cited page(s) 38, 170
V. Nannen and A.E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In M. Veloso et al., editor, *Proc. Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 975–980, 2007.

[Niehaus and Banzhaf, 2001] cited page(s) 48, 49, 52, 61
J. Niehaus and W. Banzhaf. Adaption of operator probabilities in genetic programming. In *Proc. European Conf. on Genetic Programming (EuroGP)*, pages 325–336. Springer, 2001.

[Obradovic and Srikumar, 2000] cited page(s) 16
Z. Obradovic and R. Srikumar. Constructive neural networks design using genetic optimization. *Facta Universitatis - Mathematics and Informatics Series*, 15:133–146, 2000.

[Ong and Keane, 2004] cited page(s) 161
Y.-S. Ong and A.J. Keane. Meta-Lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):99–110, 2004.

[Page, 1954] cited page(s) 83, 83, 83
E.S. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.

[Pitman and King, 2009] cited page(s) 26
M. Pitman and A. King. Engineering solutions to optimise the design of carbon-neutral tall office buildings. In *Proc. Intl. Conf. on Solutions for a Sustainable Planet*, 2009.

[P.J. Bentley et al., 2002] cited page(s) 26
P.J. Bentley et al., editor. *Creative evolutionary systems*. Morgan Kaufmann, 2002.

[P.J.M. Laarhoven et al., 1987] cited page(s) 14
P.J.M. Laarhoven et al., editor. *Simulated annealing: theory and applications*. Kluwer, 1987.

[Porumbel *et al.*, 2010] cited page(s) 25
D.C. Porumbel, J.-K. Hao, and P. Kuntz. An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. *Computers and Operations Research*, 37(10):1822–1832, 2010.

[Price *et al.*, 2005] cited page(s) 12, 23, 23, 24, 24, 25
K. Price, R. Storn, and J. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.

[Qin *et al.*, 2009] cited page(s) 62, 169
A.K. Qin, V.L. Huang, and P.N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 13(2):398–417, 2009.

[Quick *et al.*, 1996] cited page(s) 145
R.J. Quick, V.J. Rayward-Smith, and G.D. Smith. The royal road functions: Description, intent and experimentation. In *Proc. AISB Workshop on Evolutionary Computing*, volume 1143 of *LNCS*, pages 223–235. Springer, 1996.

[Quiroz *et al.*, 2007] cited page(s) 15, 26
J. Quiroz, L. Sushil, A. Shankar, and S. Dascalu. Interactive genetic algorithms for user interface design. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2007.

[Rechenberg, 1972] cited page(s) 12, 21
I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag, 1972.

[Rudolph, 1997] cited page(s) 140
G. Rudolph. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovac, 1997.

[Schwefel, 1975] cited page(s) 33, 41
H.-P. Schwefel. *Evolutionsstrategie und numerishe Optimierung*. PhD thesis, Technical University of Berlin, 1975.

[Schwefel, 1981] cited page(s) 12, 21
H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, 1981. 1995 – 2^{nd} edition.

[Schwefel, 1995] cited page(s) 32, 41
H.-P. Schwefel. *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology. Wiley Interscience, 1995.

[Semet and Schoenauer, 2006] cited page(s) 25
Y. Semet and M. Schoenauer. On the benefits of inoculation, an example in train scheduling. In M. Cattolico et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 1761–1768. ACM, 2006.

[Singh, 2006] cited page(s) 26
V. Singh. *Automatic Seismic Velocity Inversion using Multi-Objective Evolutionary Algorithms*. PhD thesis, École des Mines de Paris, 2006.

[Sinz *et al.*, 2006] cited page(s) 152
C. Sinz, N. Amla, J. Marques-Silva, E. Zarpas, D. Le-Berre, and L. Simon. SAT-Race'06, 2006. Available at: http://fmv.jku.at/sat-race-2006/, Lastly accessed on Oct 18th 2010.

[Smit and Eiben, 2009] cited page(s) 38
S.K. Smit and A.E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 399–406. IEEE, 2009.

[Smith, 1993] cited page(s) 32
R. Smith. Adaptively resizing populations: an algorithm and analysis. In S. Forrest et al., editor, *Proc. Intl. Conf. on Genetic Algorithms and their Applications*, page 653. Morgan Kaufmann, 1993.

[Smith, 1998] cited page(s) 40
J.E. Smith. *Self Adaptation in Evolutionary Algorithms*. PhD thesis, University of the West of England, Bristol, 1998.

[Smith, 2008] cited page(s) 39
J.E. Smith. Self-adaptation in evolutionary algorithms for combinatorial optimisation. In C. Cotta et al., editor, *Adaptive and Multilevel Metaheuristics*, volume 136 of *Studies in Computational Intelligence*, pages 31–57. Springer, 2008.

[Spears, 1995] cited page(s) 39, 41
W.M. Spears. Adapting crossover in evolutionary algorithms. In J.R. McDonnell et al., editor, *Proc. Conf. on Evolutionary Programming*, pages 367–384. MIT, 1995.

[Srinivas and Patnaik, 1994] cited page(s) 58
M. Srinivas and L.M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, 24(4):656–667, 1994.

[Storn and Price, 1995] cited page(s) 25
R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Intl. Computer Science Institute, 1995.

[Storn and Price, 1997] cited page(s) 12, 23, 159
R. Storn and K. Price. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[Storn and Price, 2008] cited page(s) 24, 25, 159
R. Storn and K. Price. Differential evolution homepage, 2008. Available at: http://www.ICSI.Berkeley.edu/~storn/code.html, Lastly accessed on Oct 18th 2010.

[T. Stützle et al., 2009] cited page(s) 4, 194
T. Stützle et al., editor. *Proc. Intl. Conf. on Learning and Intelligent Optimization*, volume 5851 of *LNCS*. Springer, 2009.

[T. Yu et al., 2008] cited page(s) 3, 25
T. Yu et al., editor. *Evolutionary Computation in Practice*, volume 88 of *Studies in Computational Intelligence*. Springer, 2008.

[Tan, 2007] cited page(s) 26
R.R. Tan. Hybrid evolutionary computation for the development of pollution prevention and control strategies. *Journal of Cleaner Production*, 15(10):902 – 906, 2007.

[Thathachar and Sastry, 1985] cited page(s) 53
M.A.L. Thathachar and P.S. Sastry. A class of rapidly converging algorithms for learning automata. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, SMC-15:168–175, 1985.

[Thierens, 2005] cited page(s) 52, 53, 53, 54, 61, 69, 90, 90, 100
D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In H.-G. Beyer et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 1539–1546. ACM, 2005.

[Thierens, 2007] cited page(s) 52
D. Thierens. Adaptive strategies for operator allocation. In Lobo et al. [2007], pages 77–90.

[Tuson and Ross, 1998] cited page(s) 48, 56, 57
A. Tuson and P. Ross. Adapting operator settings in genetic algorithms. *Evolutionary Computation*, 6(2):161–184, 1998.

[Ursem, 2002] cited page(s) 59
R.K. Ursem. Diversity-guided evolutionary algorithms. In J.J. Merelo-Guervós et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*, pages 462–474. Springer, 2002.

[Vajda *et al.*, 2008] cited page(s) 33, 33
P. Vajda, A.E. Eiben, and W. Hordijk. Parameter control methods for selection operators in genetic algorithms. In G. Rudolph et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*, pages 620–630. Springer, 2008.

[Verel *et al.*, 2010] cited page(s) 93
S. Verel, P. Collard, and M. Clergue. States-based evolutionary algorithm. In *Workshop Self-Star at PPSN Conference*, 2010.

[Wei *et al.*, 2008] cited page(s) 63, 157
W. Wei, C.-M. Li, and H. Zhang. Switching among non-weighting, clause weighting, and variable weighting in local search for sat. In *Proc. Intl. Conf. on Principles and Practice of Constraint Programming (CP)*, pages 313–326. Springer, 2008.

[Whitacre *et al.*, 2006] cited page(s) 6, 49, 50, 52, 61, 71
J. Whitacre, T. Pham, and R. Sarker. Use of statistical outlier detection method in adaptive evolutionary algorithms. In M. Cattolico et al., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 1345–1352. ACM, 2006.

[Whitley and Watson, 2005] cited page(s) 30
L.D. Whitley and J.P. Watson. Complexity theory and the no free lunch theorem. In E.K. Burke et al., editor, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 1–23. Springer, 2005.

[Whitley, 1994] cited page(s) 23
L.D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4:65–85, 1994.

[Wolpert and Macready, 1997] cited page(s) 30, 42
D.H. Wolpert and W.G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[Wong *et al.*, 2003] cited page(s) 48, 49, 52, 58
Y.-Y. Wong, K.-H. Lee, K.-S. Leung, and C.-W. Ho. A novel approach in parameter adaptation and diversity maintenance for GAs. *Soft Computing*, 7(8):506–515, 2003.

[Yeomans *et al.*, 2003] cited page(s) 26
J.S. Yeomans, G.H. Huang, and R. Yoogalingam. Combining simulation with evolutionary algorithms for optimal planning under uncertainty: An application to municipal solid waste management planning in the regional municipality of Hamilton-Wentworth. *Journal of Environmental Informatics*, 1(2), 2003.

[Yuan and Gallagher, 2004] cited page(s) 36
B. Yuan and M. Gallagher. Statistical racing techniques for improved empirical evaluation of evolutionary algorithms. In X. Yao et al., editor, *Proc. Intl. Conf. on Parallel Problem Solving from Nature (PPSN)*, volume 3242 of *LNCS*, pages 172–181. Springer, 2004.

[Yuan and Gallagher, 2007] cited page(s) 37
B. Yuan and M. Gallagher. Combining meta-EAs and racing for difficult EA parameter tuning tasks. In Lobo et al. [2007], pages 121–142.

[Zaharie, 2009] cited page(s) 25
D. Zaharie. Influence of crossover on the behavior of differential evolution algorithms. *Applied and Soft Computing*, 9(3):1126–1138, 2009.

[Zhang and Sanderson, 2009] cited page(s) 188
J. Zhang and A.C. Sanderson. JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009.