Controlling evolution by means of machine learning

Michèle Sebag, Caroline Ravisé, Marc Schoenauer

Controlling Evolution by means of Machine Learning

Michèle Sebag[1], Caroline Ravisé[1], Marc Schoenauer[2]
(1) : LMS, CNRS-URA 317 (2) : CMAP, CNRS-URA 756
Ecole Polytechnique, F-91128 Palaiseau
{Michele.Sebag,Caroline.Ravise,Marc.Schoenauer}@polytechnique.fr

Abstract

A safe control of evolution consists in preventing past errors of evolution to be repeated, which could be done by keeping track of the history of evolution. But maintaining and exploiting the complete history is intractable. This paper therefore investigates the use of machine learning (ML), in order to extract a manageable information from this history. More precisely, induction from examples of past trials and errors provides rules discriminating errors from trials. Such rules allow to *a priori* estimate the opportunity of next trials; this knowledge can support powerful strategies of control.

Several strategies of ML-based control are experimented on the Royal Road, a GA-deceptive and a combinatorial optimization problem. The control of mutations unexpectedly compares to that of crossovers.

1 Introduction

Control of evolution aims at keeping some balance between the exploitation and the exploration tasks devoted to evolutionary search [7]. This control involves both the selective pressure (the average number of offspring allowed for the best individual(s)) and the disruptiveness of evolution operators, which must be sufficient to discourage premature convergence [3]. Simply put, the disruptiveness of operators is evaluated from the chances for offspring not to convey the same relevant information than parents. Only boolean search spaces and crossover and mutation operators are considered throughout this paper.

Controlling the disruptiveness of crossover and mutation can be done at three levels:

- The search space can be designed as to decrease the disruptiveness of operators regarding relevant schemas; e.g. allowing don't care zones, termed *introns*, decreases the disruptiveness of both mutation and crossover [14, 12].
- Disruptiveness is directly concerned by the crossover and mutation rates. These may be adjusted by means of brute force [23] (still the most usual way), as well as through statistical estimations [9], or adaptation [8, 2], or evolution itself [13].
- Last, the effects of crossover and mutation can be adjusted by evolution itself [24, 29, 26, 4]: this only requires to include the control choices in the search space. Evolution can thereby optimize the *type* of crossover [29], or the *mask* of crossover [24], or the variance of mutation [26, 4] most suited to an individual, for free.

The control of evolution presented in this paper aims at adjusting the effects of crossover and mutation, and is originated from a common sense remark: what has been done with bad results in the past (e.g., give birth to an individual that was not to be retained in the population), needs not be repeated. Preventing evolution from repeating its past errors constitutes a safe control, i.e. a control that cannot mislead evolution. However, maintaining and exploiting the list of past errors of evolution is intractable. Therefore, the history of evolution needs be summarized

and put in some tractable way. This paper investigates the use of machine learning (ML) to this end; more precisely, induction from examples [16, 19] is used to extract rules from the past errors and trials of evolution. Such rules enable to *a priori* estimate the opportunity of next trials; by means of this estimate, several strategies of control, termed *ML-based controls*, are allowed to command the next steps of evolution.

The paper is organized as follows. Section 2 describes the automatic extraction of rules about evolution, from examples obtained through experimenting on a population or spying evolution. The use of such knowledge in order to guide the next evolution steps is discussed, and a hybrid algorithm interleaving evolution and induction is proposed.
Section 3 presents an experimental study of several ML-based controls of evolution. Besides two well-studied GA problems, the Royal Road [17, 18], and a GA-deceptive problem [31], a combinatorial optimization problem is considered, the multiple knapsack problem [11, 20].
The scope and limitations of ML-based control are discussed in section 4, with respect to related works devoted to the control of evolution [8, 24, 29] and cultural algorithms [22, 1].

2 Knowledge-Controlled Evolution

ML-based control of evolution is grounded on the following remark: evolution is made of good and bad events (crossovers and mutations). This section first shows how ML, more precisely inductive learning, can be used to characterize the classes (sets) of good and bad events, through rules induced from examples. These rules provide an *a priori* estimate of the class, good or bad, of new incoming events. How such rules can be used on-line by evolution, is then discussed.

2.1 Inductive Learning

Let us first briefly introduce inductive learning (see [16, 21] for a thorough presentation).
Examples are points of the search space which have been classified (e.g. by an expert). The goal of induction is to extract **rules** from training examples; a rule can be viewed as a schema of the search space associated to a given class. A rule R **covers** an example iff the example belongs to the schema of R. A rule **generalizes** an example, iff it covers this example and they both belong to the same class.

E_1	1	1	1	0	0	1	*good*
E_2	0	0	0	1	1	1	*good*
E_3	1	1	0	0	1	1	*bad*
E_4	1	0	0	0	1	1	*bad*
E_5	0	0	0	0	1	1	*good*
R	$\star$	$\star$	$\star$	0	1	$\star$	*bad*

Table 1 : Induction from examples

Table 1 shows some examples in $\{0,1\}^6$ representing classes *good* and *bad*, together with a rule. Induction attempts to optimize a quality function involving several features: (a) *Generality*, i.e. order of the schema in the rule; (b) *Significance*, i.e. number of examples the rule generalizes. (R generalizes Ex_3 and Ex_4), and (c) *Accuracy*, i.e. ratio between the number of examples the rule generalizes and the number of examples it covers (R covers Ex_3, Ex_4 and Ex_5; its accuracy is $2/3$).

Induction proceeds by exploring the training examples either in a top-down or in a bottom-up way. In the top-down approach [21], one builds rules or decision trees by repeatedly selecting the most discriminant genes, i.e. the gene whose value gives a maximal information regarding the class of the examples. In the bottom-up approach [16], one starts from a given example and finds out the rules that generalize this example and maximize some user-supplied quality function. Then the examples generalized by these rules are removed from the training set, and another example is considered. The learning algorithm used in this paper is a bottom-up algorithm that determines all rules maximally general with a given prescribed (user-supplied) accuracy; a constraint-based formalism allows to build such rules with a polynomial complexity [27].

Induction ultimately allows for classifying any point E in the search space: E is associated to the class of the rules covering E, (with majority vote in case of conflicts). In case where E is not covered by any rule, it is classified *unknown*.

2.2 Examples about evolution

In order to apply inductive learning, we need examples relevant to evolution, and easy to gather. The possibility investigated in this paper is to take as examples the elementary events of evolution, namely the birth of new individuals through crossover or mutation.

Description of examples. A crossover event is defined by a pair of parents and the crossover mask applied on these parents. Following Syswerda [30], a crossover c can be represented by a binary mask $(c_1, \ldots c_N)$, $c_i \in \{0, 1\}$:

$$\begin{array}{c} x_1 \ldots x_N \\ y_1 \ldots y_N \end{array} \rightarrow \begin{array}{c} x'_1 \ldots x'_N \\ y'_1 \ldots y'_N \end{array} \text{ with } \begin{array}{llll} x'_i = x_i & y'_i = y_i & \text{if } c_i = 1 \\ x'_i = y_i & y'_i = x_i & \text{otherwise} \end{array}$$

Likewise, a mutation event is defined by a parent and the mutation applied on this parent. A mutation can also be represented through a binary mask $m = (m_1, ..m_N)$, such that $x_1 \ldots x_N \rightarrow x'_1 \ldots x'_N$ with :

$$x'_i = \begin{cases} 1 - x_i & \text{if } m_i = 1 \\ x_i & \text{otherwise} \end{cases}$$

Both kinds of events can then be represented through the operator mask and the parent(s). In this paper, the **description of an example** consists of the operator mask, and optionally the parent the operator applies on (the most fit parent in the crossover case).

Then, examples must be classified in order to permit induction. It seems natural, as far as learning intends to serve control, to classify events as to whether they contribute to the current optimization task. The choice made in this paper is the following: the class of an event depends on the way the fitness of offspring compares to the fitness of parent(s). The class of an event is:

- *good* if the (best) offspring has higher fitness than the (best) parent,
- *bad* if the (best) offspring has lower fitness than the (best) parent
- *inactive* if the (best) offspring and the (best) parent have the same fitness.

Acquisition of examples. At the moment, examples are gathered through experimenting on a given population, termed *reference population*:

1. An operator mask is randomly generated according to the parameters of the evolutionary algorithm (e.g. mutation rate, n-point crossover or uniform crossover,...); inactive operators are rejected (e.g. mask 00...0);
2. One or two chromosomes (depending on whether the operator is mutation or crossover) are randomly selected in the reference population;
3. The operator is performed according to the mask and parent(s) selected. The fitness of the offspring is computed and compared to that of the parent(s). This comparison determines the class of the event, *good, bad* or *inactive*;
4. The example composed of the operator mask, optionally the (most fit) parent, and the associated class, is stored[1].

2.3 Rules about evolution

Rules are induced from the gathered examples. Only significant rules (covering more than one example) are retained.

Scope of the rules. Table 2 shows examples of 2-point crossovers together with a rule induced from these examples.

	Chromosome						*Mask*						*Class*
E_1	1	1	1	0	0	0	1	1	1	0	0	1	*good*
E_2	1	1	1	0	0	0	0	0	0	1	1	1	*good*
E_3	1	1	1	0	0	0	1	0	1	1	1	1	*bad*
E_4	1	1	1	0	0	0	0	0	1	1	1	1	*bad*
R	1	1	1	$\star$	$\star$	$\star$	$\star$	0	1	$\star$	$\star$	$\star$	*bad*

Table 2 : Induction from examples of crossover

Rule R states that : The crossover of an individual in schema $H = 111\star\star\star$ according to a crossover mask in schema $\star 01\star\star\star$, gives a *bad* result, i.e. the offspring are less fit than the parents. This can be interpreted as: don't set a crossing point between bits 2 and 3 if the parent belongs to schema H.

Rules reflect the reference population: note that R cannot be learnt before schema H is discovered, and will hardly be learnt if many individuals in the population belong to H.

ML-based control. Such rules enable to *a priori* estimate whether a next event (crossovers or mutations) is bad, good, or inactive. This estimate can accommodate several strategies of control:

- Favoring desirable events, by actuating only *good* events. However, this strategy would likely break the balance between exploration and exploitation in favor of the latter.
- Limiting the disruptiveness of operators, by rejecting bad events. This strategy of control is termed *classical*.

[1]Note that crossing over $parent_1$ with $parent_2$ according to a given crossover mask may happen to be *good*, while crossing over $parent_1$ with $parent_3$ according to the same crossover mask is *bad*. Then, if only one parent (say $parent_1$) is considered in the example description, one gets two examples with same description belonging to distinct classes, i.e. examples are inconsistent. Inconsistencies are still more likely, if the parent is omitted in the example description. Fortunately many learning algorithms can deal with a limited amount of inconsistencies, so this is not a real limitation.

- Increasing the diversity of the population, by rejecting inactive events. This strategy of control is termed *modern*.

Neither classical nor modern control actually breaks the balance between exploration and exploitation: rather, the rules delineate regions where exploration or exploitation have led to bad or null results. This allows to both biase exploration and exploitation toward other regions. ML-based control involves two kinds of cost:

- The acquisition of K examples implies at most $2 \times K$ fitness computations. The number of examples considered by induction is experimentally set to the number P of individuals in the population.
  This extra cost could be avoided if examples were gathered through spying evolution instead of experimenting on the reference population.
- The cost of induction from examples (in $\mathcal{O}(P^2 \times N)$, where N denotes the dimension of the search space, for the learner used in our experiments).

Limitations. The presented approach can fail in two ways: control may be disabled, or, even worse, misleading.

Control is disabled when induction fails to deliver significant or usable rules. This may be the case if the reference population (2.2) does not contain relevant schemas; then no trends about disruptive or inactive operators can be learnt. It may also happen that all acquired examples fall in the same class; discriminant induction then does not apply.

A much worse case is that of a misleading control, discouraging the discovery of optimal regions: control would then be properly deceptive. The deceptivity of control is to blame on the rules:
Rules may become globally erroneous, for instance if the reference population is too different from the populations undergoing control. (Similarly, the estimations made from random individuals may be not reliable as evolution goes toward regions of better and better fitness [9]).
And rules may be locally erroneous, since they generalize rather than compact the available examples. However, would the rules only compact examples, they would also allow for very few classifications, leading again to a disabled control.

Some of these limitations are addressed by the following coupling of evolution and induction.

2.4 Interleaving evolution and induction

We propose to distinguish three phases in the "game" of evolution.
The beginning of the game is characterized by a (relatively) high probability of getting offspring more fit than parents. During this phase, evolution obviously needs not be controlled. Practically, the first generations do not undergo any control.
ML-control then waits until relevant schemas appear, so that significant rules can be learnt. This prevents the first risk of disabled control.

The middle of the game is characterized by a high probability of getting offspring less fit than parents. During this phase, relevant schemas likely have emerged, but not yet crowded the population. The main concern here is to limit the disruptiveness of operators, which can be done through a classical ML-control (discarding disruptive operations).

The end of the game is characterized by a (relatively) high probability of getting offspring as fit as the parents. During this phase, the population is getting homogeneous. A main concern

would then be to preserve the diversity of the population, which can be done through a modern ML-control (discarding inactive operations).

The deceptivity of the control is partially prevented through periodically updating the rules. Every M generations, the reference population being set to the current population, new examples are gathered.
If these new examples do not enable induction (characterized as, the fraction of examples in the majority class exceeds some user-supplied threshold D, with $D < 100\%$) then control is disabled. The next M generations undergo darwinian evolution.
Otherwise, if the age of evolution is qualified as "middle of the game" (characterized as, the fraction of inactive examples is less than some user-supplied threshold I, with $I \leq D$), then a classical ML-control is performed in order to limit disruptiveness during the next M generations, termed *classical* period.
Otherwise, the age of evolution is qualified as "end of the game" and a modern control is performed in order to preserve diversity during the next M generations, termed *modern* period.

The number M of successive generations controlled through the same rules (in case of classical or modern periods) is experimentally set to 3: a large value of M may lead to a deceptive control in the last generations of the period; and small values of M increase the overall cost of controlled evolution, without definite benefits.

ML-controlled evolution can then be viewed as a mixture of darwinian, classical and modern periods. The occurrences of darwinian periods are governed by parameter D: as D decreases, the majority class tends to be represented by more than $D\%$ of the examples. Similarly, the occurrences of modern periods are governed by parameter I.

3 Experimental Validation

The aim of the presented experimentations is twofold. The behavior of an ML-controlled evolution is studied through varying values of D and I, which allows to compare different mixtures of darwinian, classical and modern periods. Besides, this approach gives a unique opportunity to study the roles respectively devoted to mutation and crossover, by comparing what happens when mutations only, then crossovers only, are controlled.

Three problems are considered: the Royal Road problem [17], a GA-deceptive problem [31], and a combinatorial optimization problem [11].

3.1 Experimental settings

The evolutionary algorithm is a standard GA [7] with bit-string encoding, roulette wheel selection with fitness scaling, two-points crossover at a rate of 0.6 with both offspring replacing the parents. Mutation is performed at a rate of 0.005. The evolution stops after 15000 fitness evaluations. Fitness scaling is used with a selective pressure 1.2 or 2. The size of the population is 25.
The ML algorithm used, called *CBI* for *Constraint-Based Induction*, is described in detail in [27].
Acquisition of examples and induction are performed every 3 generations, the first three generations being darwinian.

The results are given in terms of percentage of success over 15 independent runs (success is intended as hitting the maximum, known for all considered problems). The dynamics of evolution is visualized by plotting the average best fitness (over 15 runs) obtained for a given number of fitness calculations. These include of course the extra calculations required by ML-control.

Several evolution schemes are compared: A classical GA first (legend **GA**) that serves as reference. Then two GAs with a GA-based control of crossover are experimented: the crossover control described by Spears [29] (legend **Sp**) where an additional bit commands the kind of crossover, uniform or 2-point, to be applied on the individual ; and the crossover control described by Schaffer and Morishima [24] (legend **S-M**) where individuals are augmented by the crossover mask to be applied on them. Last, four schemes of ML-based control are experimented:

- Control applies on crossovers only, and the underlying rules are induced from only examples of crossovers (legend **X-X**).
- Control applies on crossovers and mutations, and the underlying rules are induced from only examples of crossovers (legend **X-XM**).
- Control applies on mutations only, and the underlying rules are induced from only examples of mutations (legend **M-M**).
- Control applies on crossovers and mutations, and the underlying rules are induced from only examples of mutations (legend **M-XM**).

The fact that a given kind of operations can be controlled through rules learnt from operations of another kind, can be justified as follows. Mutating an individual x through a mutation mask m can be viewed as crossing-over x with its complementary $\neg x$ through crossover mask $c = m$. (See also [10]). This implies that rules learnt from mutations enable a too severe control of crossovers (x is usually crossed with an individual nearer to x than $\neg x$; and crossover gives two offspring), and reciprocally, rules learnt from crossovers enable a loose control of mutations. In both cases, the control is still worth trying.

3.2 The Royal Road

The Royal Road problem was conceived by Holland and Mitchell [17] to study into details the combination of features most adapted to GA search (laying a *Royal Road*). An analysis of the unexpected difficulties of this problem can be found in [18, 6].

Table 3 shows the results obtained on the Royal Road problem, modified as in [18], for selective pressure 1.2 and 2. Results indicated for ML-controlled evolutions correspond to the average of the results obtained for $D = 95\%$ and I in $\{50\%, 67\%, 95\%\}$ (see Table 4 for detailed results).

sel. press.	GA	Controlled GA					
		GA-based Ctrl		ML-based Ctrl			
		Sp	*S-M*	*M-M*	*M-MX*	*X-X*	*X-MX*
1.2	80	83	73	93	95	55	44
2	93	100	100	100	100	95	84

Table 3: The Royal Road. Percentage of success of GA with and without control

Obviously, there is few room for control when the classical GA is efficient, i.e. for selective pressure 2. But globally, the ML-control built from examples of crossovers ($X-X$ and $X-XM$) is harmful, and in any case much less efficient than other GA-based controls of crossover.
In opposition, the ML-control built from examples of mutations ($M-M$ and $M-MX$) reaches the same results than GA-based control for selective pressure 2., and significantly supersedes other evolution schemes for selective pressure 1.2.

The influence of parameters D (commanding the occurrences of darwinian periods) and I (commanding the occurrences of modern periods), is shown in Table 4, and discussed in 3.5.

ML-control	$D = I = 50\%$	$D = 67\%$		$D = 95\%$		
		$I = 50\%$	$I = 67\%$	$I = 50\%$	$I = 67\%$	$I = 95\%$
$M-M$	73	60	93	93	100	87
$M-MX$	47	80	73	100	87	100
$X-X$	33	47	27	60	53	53
$X-XM$	40	73	80	33	47	53

Table 4: The Royal Road. Detailed results of ML-based control. Selective pressure = 1.2

3.3 A GA-deceptive problem

An elementary deceptive fitness is defined on $\Omega = \{0,1\}^3$, by F(x) = 3 if x = 111; F(x) = 2 for x in $0\star\star$, and F(x) = 0 otherwise. The deceptive problem we considered is composed of 10 concatenated elementary deceptive problems [31].

The percentages of success are indicated in Table 5. Results of ML-controlled schemes are averaged over 45 runs, corresponding to $D = 95\%$ and I in {50%, 67%, 95%} (detailed results for selective pressure 1.2 are given in Table 6).

sel. press.	GA	Controlled GA					
		GA-based Ctrl		ML-based Ctrl			
		Sp	*S-M*	*M-M*	*M-MX*	*X-X*	*X-MX*
1.2	80	83	87	93	93	61	42
2	93	90	87	100	100	58	49

Table 5: A GA-deceptive problem. Percentage of success of GA with and without control

ML-control	$D = I = 50\%$	$D = 67\%$		$D = 95\%$		
		$I = 50\%$	$I = 67\%$	$I = 50\%$	$I = 67\%$	$I = 95\%$
$M-M$	87	73	93	100	80	100
$M-MX$	93	93	80	93	87	100
$X-X$	40	53	40	73	57	53
$X-XM$	27	7	47	27	13	87

Table 6: A GA-deceptive problem. Detailed results of ML-based control. Sel. pressure = 1.2

3.4 The multiple knapsack problem

The multiple knapsack problem [11] is a combinatorial optimization problem defined as follows:
– Let P knapsacks have respective capacities $c_1..c_P$,
– Let $\mathcal{O}$ denote a set of N objects, whose cost is respectively $p_1.., p_N$,

– Let $w_{i,j}$ be the overall dimension of object i regarding knapsack j;

Determine a subset of $\mathcal{O}$, noted $X = x_1, ..x_N$, with x_i boolean, that is feasible, i.e. satisfies the constraints relative to the maximal capacities of all knapsacks, and maximizes the overall profit:

$$\text{Max } \{\sum_{i=1}^{N} p_i.x_i \ ; \ \forall j = 1..P, \ \sum_{i=1}^{N} w_{i,j}x_i < c_j.\}$$

Much attention has been paid to evolutionary constrained optimization [15, 25]. A usual heuristics consists in reducing the fitness of non feasible individuals by a penalty term. We considered a multiplicative penalization:

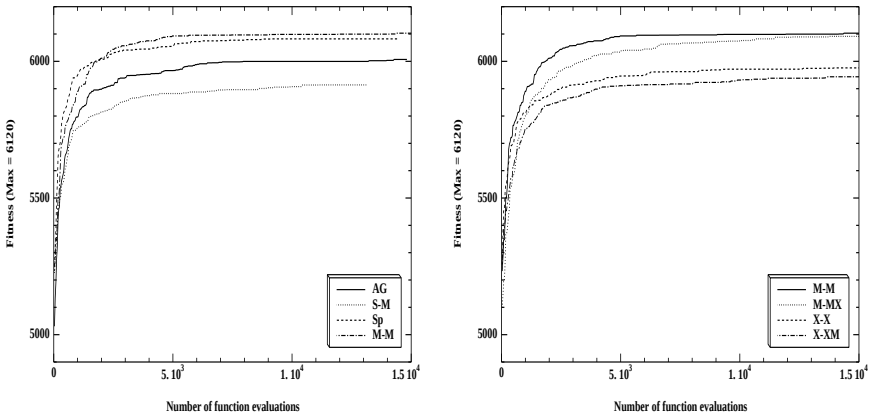
$$F(X) = \begin{cases} \sum_{i=1}^{N} p_i x_i & \text{if } X \text{ is feasible} \\ \\ \frac{r}{2} \sum_{i=1}^{N} p_i x_i & \text{if } r \text{ is the percentage of satisfied constraints} \end{cases}$$

Table 7 reports the results obtained on the fourth problem defined by Petersen [20], with $N = 20$ and $P = 10$. Similar results are obtained on the other data sets. Again, results indicated for ML-based controls are averaged on several values of D and I, which are detailed in Table 8.

sel. press.	GA	Controlled GA					
		GA-based Ctrl		ML-based Ctrl			
		Sp	*S-M*	*M-M*	*M-MX*	*X-X*	*X-MX*
1.2	13	20	0	29	28	9	17
2	0	0	0	19	11	0	4

Table 7: The knapsack problem. Percentage of success of GA with and without control

The dynamics of evolution (Figure 1) shows that ML-based control of mutations reaches sooner better solutions.



(a) With and without control (b) ML-controls

Figure 1 : The knapsack problem. Dynamics of evolution. Sel. pressure 1.2. ; $D = I = 95\%$

ML-control	$D = I = 50\%$	$D = 67\%$		$D = 95\%$		
		$I = 50\%$	$I = 67\%$	$I = 50\%$	$I = 67\%$	$I = 95\%$
$M - M$	27	20	20	40	40	27
$M - MX$	47	13	33	20	27	27
$X - X$	7	13	7	7	13	7
$X - XM$	13	33	-	13	27	13

Table 8: The knapsack problem. Detailed results of ML-based control. Selective pressure = 1.2

3.5 Remarks

On these three artificial problems, the ML-based control built from examples of mutations significantly and consistently improves on classical GA and other GA-based controls. In the meanwhile, the ML-based control built from examples of crossovers shows disastrous.

In the Royal Road and the GA-deceptive problems, the best option is that of a classical permanent control ($D = I = 95\%$), preventing disruptive mutations only. In the combinatorial optimization problem, the best control is also classical, but prevents disruptive crossovers as well as disruptive mutations.

4 Discussion

From the above results, it appears that controlling the disruptiveness of mutation can be more effective than that of crossover. After an attempt to explain this fact, we focus on the ML aspects of the presented control, with respect to some related works.

4.1 Controlling mutation

The disruptiveness of crossovers seems at first to deserve more attention than that of mutations, since the crossover rate is one or several orders of magnitude greater than the mutation rate [14, 29, 24, 3, 28].

However, the homogenization of population can efficiently counteract the disruptiveness of crossovers, and does so in the end of evolution. In the meanwhile, *nothing can ever counteract the disruptiveness of mutations*, but control. A controlled mutation then appears a powerful means to prevent the loss of near-optimal schemas in the end of evolution. This way, it improves the "memory" of evolution.

Such effect was so far expected from selection only: the loss of good individuals can also be prevented through elitist replacement or strong selection.

If the memory of evolution is too efficient, due to either controlled mutation, elitism or strong selection, this favors premature convergence. But controlled mutation leaves less room than selection to premature convergence: First, mutation tends to increase the diversity of a homogeneous population; in opposition, selection and elitism always decrease this diversity. Second, controlled mutation tends to increase the number of active bits in a mutation mask[2], thereby increasing the mutation rate.

4.2 A ML approach

The presented approach involves three points.

First, we formalize the goal of control in terms of what should be avoided (disruptiveness or loss of diversity); in opposition, previous approaches of control rather attempt to determine what should be done [29, 24, 26]. We claim that a negative control (made of inhibitions), is safer than a positive one (made of recommendations). On one hand, suitable recommendations are

[2]The first mutation examples have very few bits active. By rejecting the schemas containing some of them, mutation masks are gradually biases toward regions with more and more active bits.

outnumbered by suitable inhibitions, especially in the end of evolution. On the other hand, we know part of the suitable inhibitions (e.g., the past errors of evolution) while we know nothing like *a priori* suitable recommendations for non-trivial problems.

Second, we express control within the formalism of logical rules. Previous approaches aim at controlling evolution either at a global level (e.g., operator rates [8, 9, 13]) or at the level of each individual (e.g., suited type or mask of operators [29, 24, 26]). Rules offer a tractable and compact way to handle schemas of operators: the rule-based control applies on the whole population, and can still take into account the topology of the search space (e.g. don't mutate a given bit; mutate simultaneously a set of bits,...).

Last, we propose a procedure to extract the rules underlying control: inductive learning from examples. A further perspective of research deals with setting the rules through evolution itself: according to the fans of *Nature Only*, evolution can handle all choices pertaining to the representation space, and does so in an optimal way. Experimentations will tell whether rules of control are better adjusted by evolution, or faster extracted by an *ad hoc* external algorithm.

4.3 Cultural algorithms

The presented approach can be viewed as a particular type of cultural algorithm [22, 1]. Cultural algorithms involve both the space of individuals, and the space of schemas of individuals, termed *beliefs*. A belief is built by generalization of individual experience in the population; a lattice of shared beliefs is then built, which allows via a communication protocol, to biase the next operations to be applied on individuals.

Both beliefs and control rules encapsulate some knowledge about evolution, that is automatically extracted; this knowledge is similarly used to guide the next evolution steps.

The main difference lies in the way this knowledge is updated. The update of the beliefs is mainly a cumulative mechanism, as it proceeds by generalization of individual experiences. Furthermore, these experiences themselves are biased by the control. The risk is then to gradually validate some erroneous generalizations; simply put, this mechanism is apt to build prejudices as well as beliefs.

In opposition, we propose a quite rough update mechanism: every M generations, rules are learnt anew. Furthermore, they are learnt from *experiences which are not biased by the control*. In other words, the "memory" of the control is erased. This gives opportunity to get rid of old prejudices (erroneous rules). Other prejudices may be introduced, but long lasting prejudices are less likely to distort the control and the course of evolution.

5 Conclusion and Perspectives

This work is oriented toward building and using an explicit memory of evolution, expressed through rules. The rule formalism allows for handling knowledge that is both general (related to the whole population) and specific (related to particular genes or sets of genes).

Rules are used to express the significant trends regarding disruptive and inactive operations; these are periodically built by induction from experimentations conducted on the current population. These rules enable to *a priori* estimate the effects of further operations. Two modes of control are then possible: Classical control aims at preventing disruptiveness, through rejecting disruptive operations. Modern control aims at increasing the diversity of the population through

rejecting inactive operations.

An hybrid evolution, interleaving darwinian periods and periods undergoing a classical or modern control, is described. The strategy of control is inspired from the analogy between games and evolution. Evolution is darwinian during the beginning phase, then it undergoes a classical control during the middle of the game, and it undergoes a modern control during the end of the game. Indicators of transition are suggested.

This approach addresses the control of both crossovers and mutations. Quite unexpectedly, experimentations demonstrate the control of mutations to be much more efficient than that of crossovers, in spite of the fact that the crossover rate is much greater than the mutation rate. A tentative explanation is given (4.1).

These results suggest several avenues for further research.

First, the strategy of control could be defined in a more flexible way. For instance, the description of an individual could include the mode of control, classical, modern or darwinian, to be applied on this individual. Evolution would thereby optimize for free the strategy of control, *a la* Spears [29].

Second, ML-control will be experimented in the evolutionary programming frame. Further experimentations will be conducted to understand the potentialities of controlled mutation, and see to what extent it constitutes an alternative to crossover [5, 10].

Third, this approach will be extended to handle real-valued search spaces. The feasibility of this extension is straightforward: Mutation and crossover can be given a mask representation with masks in $[-1, 1]^N$ [15]. Many learners allow to extract rules (hyper rectangles) from examples in $\mathbb{R}^N$. But unexpected problems are likely to appear with experimentations.

References

[1] M.J. Cavaretta. Using a cultural algorithm to control genetic operators. In A.V. Sebald and L.J. Fogel, editors, 3^{rd} *Annual Conference on Evolutionary Programming*, pages 158–166, World Scientific, 1994.

[2] L. Davis. Adapting operator probabilities in genetic algorithms. In *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*, pages 61–69, Morgan Kaufmann, 1989.

[3] K.A. DeJong and W.M. Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Artificial Intelligence*, 5:1–26, 1992.

[4] D. B. Fogel, L. J. Fogel, W. Atmar, and G. B. Fogel. Hierarchic methods of evolutionary programming. In L. J. Fogel and W. Atmar, editors, *Proceedings of the First Annual Conference on Evolutionary Programming*, pages 175–182, 1992.

[5] D.B. Fogel and L.C. Stayton. On the effectiveness of crossover in simulated evolutionary optimization. *BioSystems*, 32:171–182, 1994.

[6] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithms : Some anomalous results and their explanation. *Machine Learning*, pages 285–319, 1993.

[7] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison Wesley, 1989.

[8] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-16, 1986.

[9] J. J. Grefenstette. Virtual genetic algorithms: First results. Technical Report AIC-95-013, Navy Center for Applied Research in Artificial Intelligence, February 1995.

[10] T. Jones. Crossover, macromutation and population-based search. In L. Eschelman, editor, *Proceedings of 6^{th} International Conference on Genetic Algorithms*, pages 73–80, Morgan Kaufmann, 1995.

[11] S. Khuri, T. Bäck, and J. Heitkötter. The 0/1 multiple knapsack problem and genetic algorithms. In *Proceedings of the ACM Symposium of Applied Computation*, 1994.

[12] J. R. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Evolution*. MIT Press, Massachussetts, 1994.

[13] M.A. Lee and H. Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. In S. Forrest, editor, *Proceedings of 5^{th} International Conference on Genetic Algorithms*, pages 76–83, Morgan Kaufmann, 1993.

[14] J. R. Levenick. Inserting introns improves genetic algorithm success rate : Taking a cue from biology. In R.K. Belew and L.B. Booker, editors, *Proceedings of 4^{th} International Conference on Genetic Algorithms*, pages 123–127, Morgan Kaufmann, 1991.

[15] Z. Michalewicz. *Genetic Algorithms+Data Structures=Evolution Programs*. Springer Verlag, 1992.

[16] R.S. Michalski. A theory and methodology of inductive learning. In R.S Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning : an artificial intelligence approach*, volume 1. Morgan Kaufmann, 1983.

[17] M. Mitchell, S. Forrest, and J.H. Holland. The royal road for genetic algorithms : Fitness landscapes and ga performance. In F. J. Valera and P. Bourgine, editors, *Proceedings of the First European Conference on Artificial Life-93*, pages 245–254. MIT Press, 1993.

[18] M. Mitchell and J.H. Holland. When will a genetic algorithm outperform hill-climbing ? In S. Forrest, editor, *Proceedings of the 5^{th} International Conference on Genetic Algorithms* page 647, 1993.

[19] T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[20] C.C. Petersen. Computational experience with variants of the balas algorithm applied to the selection of r & d projects. *Management Science*, 13:736–750, 1967.

[21] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[22] R.G. Reynolds. An introduction to cultural algorithms. In A.V. Sebald and L.J. Fogel, editors, 3^{rd} *Annual Conference on Evolutionary Programming*, World Scientific, pages 131–139, 1994.

[23] J. D. Schaffer, R. A. Caruana, L. Eschelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In J. Schaffer, editor, *Proceedings of the 3^{rd} International Conference on Genetic Algorithms*, pages 51–60, Morgan Kaufmann, 1989.

[24] J.D. Schaffer and A. Morishima. An adaptive crossover distribution mechanism for genetic algorithms. In *ICGA2*, pages 36–40, Morgan Kaufmann, 1987.

[25] M. Schoenauer and S. Xanthakis. Constrained ga optimization. In Forrest S., editor, *Proceedings of 4^{th} International Conference on Genetic Algorithms*, pages 573–580, Morgan Kaufmann, 1993.

[26] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, New-York, 1981.

[27] M. Sebag. Using constraints to building version spaces. In L. De Raedt and F. Bergadano, editors, *Proceedings of ECML-94, European Conference on Machine Learning*, Springer Verlag, 1994.

[28] M. Sebag and M. Schoenauer. Controlling crossover through inductive learning. In H.P. Schwefel, editor, *Proceedings of PPSN-94, Parallel Problem Solving from Nature*. Springer-Verlag, LNCS 866, 1994.

[29] W. M. Spears. Adapting crossover in a genetic algorithm. In R. K. Belew and L. B. Booker, editors, *Proceedings of the 4^{th} International Conference on Genetic Algorithms*, Morgan Kaufmann, 1991.

[30] G. Syswerda. Uniform crossover in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the 3^{rd} International Conference on Genetic Algorithms*, pages 2–9, Morgan Kauffman, 1989.

[31] D. Whitley. Fundamental principles of deception in genetic search. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991.