An Advanced Evolution Should Not Repeat its Past Errors

Caroline Ravisé and Michèle Sebag
LMS, CNRS URA 317, École Polytechnique, 91128 Palaiseau, FRANCE
LRI, CNRS URA 410, Université Paris-Sud, 91405 Orsay, FRANCE
Caroline.Ravise@polytechnique.fr, Michele.Sebag@polytechnique.fr

Abstract

A safe control of genetic evolution consists in preventing past errors of evolution from being repeated. This could be done by keeping track of the history of evolution, but maintaining and exploiting the complete history is intractable.

This paper investigates the use of machine learning (ML), in order to extract manageable information from this history. More precisely, induction from examples of past trials and errors provides rules discriminating errors from successful trials. Such rules allow to *a priori* estimate the desirability of future trials; this knowledge can support powerful control strategies.

Several strategies of ML-based control are applied to a genetic algorithm, and tested on the Royal Road, a GA-deceptive, and a combinatorial optimization problem. Comparing mutation control with crossover control yields unexpected results.

1 INTRODUCTION

Control of evolution aims at keeping some balance between the exploitation and exploration tasks devoted to evolutionary search (Goldberg 1989). This control involves both the selective pressure[1] and the disruptiveness of evolution operators[2], which must be sufficient to respectively ensure convergence, and discourage premature convergence (De Jong & Spears 1992). Only boolean search spaces and crossover and mutation operators are considered throughout this paper.

[1]The average number of offspring allowed for the best individual(s)

[2]The chances for offsprings not to convey the same relevant information as parent(s).

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

- The search space can be designed so 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 (Levenick 1991, Koza 1994).
- Disruptiveness is directly affected by the crossover and mutation rates. These may be adjusted by means of brute force (Schaffer et al. 1989) (still the most usual method), through statistical estimates (Grefenstette 1995), adaptation (Grefenstette 1986, Davis 1989), or evolution itself (Lee & Takagi 1993).
- Finally, the effects of crossover and mutation can be adjusted by evolution itself (Schaffer & Morishima 1987, Spears 1991, Schwefel 1981, Fogel et al. 1992): this only requires to include control choices in the search space. Evolution can thereby optimize for free the *type* of crossover (Spears 1991), or the *mask* of crossover (Schaffer & Morishima 1987), or the variance of mutation (Schwefel 1981, Fogel et al. 1992) most suited to an individual.

The control of evolution presented in this paper aims at adjusting the effects of crossover and mutation, and is derived 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), should 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 must be summarized and saved into a tractable form. This paper investigates the use of machine learning (ML) to this end; more precisely, induction from examples (Michalski 1983, Mitchell 1982) is used to extract rules from the past errors and trials of evolution. Such rules allow to *a priori* estimate the desirability of future trials; by means of this estimate, several control strategies, termed *ML-based controls*, are allowed to direct 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 (Mitchell et al. 1993, Mitchell & Holland 1993) and a GA-deceptive problem (Whitley 1991), a combinatorial optimization problem is considered: the multiple knapsack problem (Khuri et al. 1994, Petersen 1967). The scope and limitations of ML-based control are discussed in section 4, with respect to related work devoted to the control of evolution (Grefenstette 1986, Schaffer & Morishima 1987, Spears 1991) and cultural algorithms (Reynolds 1994, Cavaretta 1994).

2 KNOWLEDGE-CONTROLLED EVOLUTION

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

2.1 INDUCTIVE LEARNING

Let us first briefly introduce inductive learning (see (Michalski 1983, Quinlan 1986) 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.
Table 1 shows some examples in $\{0,1\}^6$ belonging to 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 E_3 and E_4), and (c) *Accuracy*, i.e. ratio of generalized to covered examples (R covers E_3, E_4 and E_5; its accuracy is 2/3).

Table 1 : Induction from examples

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*

Induction proceeds by exploring the training examples either in a top-down or in a bottom-up fashion. In the top-down approach (Quinlan 1986), one builds rules or decision trees by repeatedly selecting the most discriminant gene, i.e. the gene whose value gives maximal information regarding the class of the examples. In the bottom-up approach (Michalski 1983), one starts from a given example and finds out the rules that generalize this example and maximize some user-supplied quality function. The examples generalized by these rules are then 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 disjunctive formalism allows to characterize such rules with polynomial complexity (Sebag 1996).

Induction ultimately allows for classifying any point E in the search space: E is associated with 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 to these parents. Following Syswerda (1989), a crossover c can be represented by a binary mask $(c_1, \ldots c_N)$, $c_i \in \{0,1\}$:

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

Likewise, a mutation event is defined by a parent and the mutation applied to 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 by 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 to (only the fittest parent in the crossover case).

Examples must then be classified in order to permit induction. It seems natural, as far as learning intends to serve control, to classify events according 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 experiments 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 applied 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 (fittest) parent, and the associated class, is stored[3].

2.3 RULES ABOUT EVOLUTION

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

[3]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 identical descriptions belonging to distinct classes, i.e. inconsistent examples. Inconsistencies are even more likely, if the parent is omitted from the example description. Fortunately many learning algorithms can deal with a limited amount of inconsistencies, so this is not a real limitation.

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

Table 2 : Induction from examples of crossover

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

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 offsprings 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: 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 future event (crossover or mutation) is bad, good, or inactive. This estimate can accommodate several control strategies:

- 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 control strategy is termed *classical*.
- Increasing the diversity of the population, by rejecting inactive events. This control strategy 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 biase both 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 (Sebag 1996).

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 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 unreliable as evolution goes toward regions of better and better fitness (Grefenstette 1995)).
Rules may also be locally erroneous, since they generalize rather than compact the available examples. However, should the rules only compact examples, they would also allow for very few classifications, thereby leading 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 classical ML-control (discarding disruptive operations).

The *end of the game* is characterized by a (relatively) high probability of getting offspring as fit as their 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 modern ML-control (discarding inactive operations).

The deceptivity of control is partially prevented through periodically updating the rules. Every M generations, the reference population is set to the current population and new examples are gathered. If these new examples do not enable induction (characterized by: 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 by: the fraction of inactive examples is less than some user-supplied threshold I, with $I \leq D$), then 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 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 while 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 experiments is twofold. The behavior of ML-controlled evolution is studied through varying values of D and I, which allows to compare different mixtures of darwinian, classical and modern periods. In addition, 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 (Mitchell et al. 1993), a GA-deceptive problem (Whitley 1991), and a combinatorial optimization problem (Khuri et al. 1994).

3.1 EXPERIMENTAL SETTINGS

The evolutionary algorithm is a standard GA (Goldberg 1989) with bit-string encoding, roulette wheel selection with fitness scaling, two-points crossover at a

rate of 0.6 with both offsprings replacing the parents. Mutation is performed at a rate of 0.005. The evolution stops after $15,000$ fitness evaluations. Fitness scaling is used with a selective pressure of 1.2 or 2. The size of the population is 25.
The ML algorithm used, called *DiVS* for *Disjunctive Version Space*, is described in detail in (Sebag 1996). 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 defined 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 (Spears 1991) (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 (Schaffer & Morishima 1987) (legend **S-M**) where individuals are augmented by the crossover mask to be applied to them. Last, four schemes of ML-based control are experimented:

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

The fact that a given kind of operation 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 Jones 1995). This implies that rules learnt from mutations enable an excessively severe control of crossovers (x is usually crossed with an individual nearer to x than $\neg x$; and crossover gives two offsprings), and conversely, 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 designed by Holland and Mitchell (Mitchell et al. 1993) to study in detail the combination of features most adapted to GA search (laying a *Royal Road*). An analysis of unexpected difficulties of this problem can be found in (Mitchell & Holland 1993, Forrest & Mitchell 1993).

Table 3 shows the results obtained on the Royal Road problem, modified as in (Mitchell & Holland 1993), for selective pressure (*s.p.*) 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).

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

		Controlled GA					
		GA Ctrl		ML Ctrl			
s.p.	GA	*Sp*	*S-M*	M-M	M-MX	X-X	X-XM
1.2	80	83	73	93	95	55	44
2	93	100	100	100	100	95	84

Obviously, there is little 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 contrast, the ML-control built from examples of mutations ($M - M$ and $M - MX$) achieves the same results as GA-based control for selective pressure 2., and significantly outperforms other evolution schemes for selective pressure 1.2.

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

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

D	50%	67%		95%		
I	50%	50%	67%	50%	67%	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

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 (Whitley 1991).

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).

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

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

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

D	50%	67%		95%		
I	50%	50%	67%	50%	67%	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

3.4 THE MULTIPLE KNAPSACK PROBLEM

The multiple knapsack problem (Khuri et al. 1994) 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, the cost of which 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 (Schoenauer & Xanthakis 1993). A usual heuristic consists in reducing the fitness of non feasible individuals by a penalty term. We considered a multiplicative penalization:

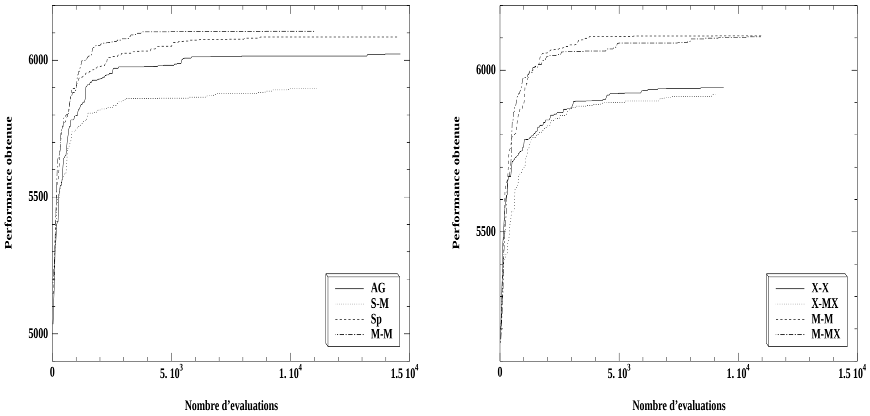
$$F(X) = \begin{cases} \sum_{i=1}^{N} p_i x_i & \text{if } X \text{ is feasible} \\ \frac{r}{2} \sum_{i=1}^{N} p_i x_i & \text{otherwise} \end{cases}$$

where r is the percentage of satisfied constraints

Table 7 reports the results obtained on the fourth problem defined by Petersen (1967), 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.

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

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



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

Figure 1 : The knapsack problem. Dynamics of evolution.

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

D	50%	67%		95%		
I	50%	50%	67%	50%	67%	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

The dynamics of evolution shows that ML-based control of mutations reaches sooner better solutions (Figure 1: selective pressure is 1.2. ; $D = I = 95\%$).

3.5 REMARKS

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

In the Royal Road and the GA-deceptive problems, the best option is that of permanent classical 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 mutations can be more effective than that of crossovers. 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 higher than the mutation rate (Levenick 1991, Spears 1991, Schaffer & Morishima 1987, De Jong & Spears 1992, Sebag & Schoenauer 1994). However, population homogenization can efficiently counteract the disruptiveness of crossovers, and more so in the end of evolution. On the other hand, *nothing can ever counteract the disruptiveness of mutations*, but control. Controlled mutation thus appears as 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 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 contrast, selection and elitism always decrease this diversity. Second, controlled mutation tends to increase the number of active bits in a mutation mask[4], thereby increasing the mutation rate.

4.2 A ML APPROACH

The presented approach involves three key points. First, we formalize the goal of control in terms of *what should be avoided* (disruptiveness or loss of diversity); previous approaches of control typically attempt to determine what should be done (Spears 1991, Schaffer & Morishima 1987, Schwefel 1981). We claim that a

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

negative control (made of inhibitions), is safer than a positive one (made of recommendations). On the one hand, suitable recommendations are 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: Grefenstette 1986, Grefenstette 1995, Lee & Takagi 1993) or at the level of each individual (e.g., suited type or mask of operators: Spears 1991, Schaffer & Morishima 1987, Schwefel 1981). Rules offer a tractable and compact way to handle schemas of operators: rule-based control applies to 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,...).

Finally, we propose a procedure to extract the rules underlying control: inductive learning from examples[5]. 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 control rules are better adjusted by evolution, or faster extracted by an *ad hoc* external algorithm.

5 CONCLUSION AND PERSPECTIVES

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

Rules are used to express significant trends regarding disruptive and inactive operations; these are periodically built by induction from experiments 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, by rejecting disruptive operations. Modern control aims at increasing population diversity by rejecting inactive operations.

A hybrid evolution scenario, interleaving darwinian periods and periods undergoing a classical or modern control, is described. The control strategy is inspired from the analogy between games and evolution. Evolution is darwinian during the beginning phase, then it undergoes classical control during the middle of the game, and 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, experiments 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 higher than the mutation rate. A tentative explanation is given (4.1).

These results suggest several avenues for further research.
First, the control strategy 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 to this individual. Evolution would thereby optimize the control strategy for free, *a la* Spears (1991).
Further experimentations will also be conducted to understand the potentials of controlled mutation, and see to what extent it constitutes an alternative to crossover (Fogel & Stayton 1994, Jones 1995).
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 $[-1,1]^N$. Many learners are able to extract rules (hyper rectangles) from examples in $\mathbb{R}^N$. But unexpected problems will likely emerge from experimentations...

Acknowledgments

We heartfully thank Marc Schoenauer from CMAP, Ecole Polytechnique, who took part in the beginnings of *Advanced Evolution*, for his invaluable comments about the present paper.

References

Cavaretta, M.J. (1994). Using a cultural algorithm to control genetic operators. In A.V. Sebald & L.J. Fogel (ed.), 3^{rd} *Conference on Evolutionary Programming*, World Scientific, 158-166.

Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In 3^{rd} *ICGA*, 61-69, Morgan Kaufmann.

De Jong, K.A. & Spears, W.M. (1992). A formal analysis

[5] This could be viewed in the line of cultural algorithms, that similarly build and use "beliefs" to guide evolution (Reynolds 1994, Cavaretta 1994). However, a significant difference lies in the update mechanism: in cultural algorithms, new beliefs are built on the basis of experiments *biased according to old beliefs*. The risk is then to gradually validate some erroneous generalizations; simply put, this mechanism is apt to build prejudices as well as beliefs. In contrast, in our approach, the "memory" of control is erased, in the sense that control rules are learnt anew every M generations. This gives opportunities 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.

of the role of multi-point crossover in genetic algorithms. *Artificial Intelligence*, 5:1-26.

Fogel, D.B., Fogel, L.J., Atmar, W. & Fogel, G.B. (1992). Hierarchic methods of evolutionary programming. In L.J. Fogel & W. Atmar (ed.), 1^{st} *Conference on Evolutionary Programming*, 175-182.

Fogel, D.B. & Stayton, L.C. (1994). On the effectiveness of crossover in simulated evolutionary optimization. *BioSystems*, 32:171-182.

Forrest, S. & Mitchell, M. (1993). What makes a problem hard for a genetic algorithms : Some anomalous results and their explanation. *Machine Learning*, 285-319.

Goldberg, D.E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison Wesley.

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

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

Jones, T. (1995). Crossover, macromutation and population-based search. In L. Eschelman (ed.), 6^{th} *ICGA*, 73-80, Morgan Kaufmann.

Khuri, S., Bäck, T. & Heitkötter, J. (1994). The 0/1 multiple knapsack problem and genetic algorithms. In *ACM Symposium of Applied Computation*.

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

Lee, M.A. & Takagi, H. (1993). Dynamic control of genetic algorithms using fuzzy logic techniques. In S. Forrest (ed.), 5^{th} *ICGA*, 76-83, Morgan Kaufmann.

Levenick, J.R. (1991). Inserting introns improves genetic algorithm success rate : Taking a cue from biology. In R.K. Belew & L.B. Booker (ed.), 4^{th} *ICGA*, 123-127, Morgan Kaufmann.

Michalski, R.S. (1983). A theory and methodology of inductive learning. In R.S Michalski, J.G. Carbonell, & T.M. Mitchell (ed.), *Machine Learning : an artificial intelligence approach*. Morgan Kaufmann.

Mitchell, M., Forrest, S. & Holland, J.H. (1993). The royal road for genetic algorithms : Fitness landscapes and ga performance. In F.J. Valera & P. Bourgine (ed.), 1^{st} *ECAL*, 245-254. MIT Press.

Mitchell, M. & Holland, J.H. (1993). When will a genetic algorithm outperform hill-climbing ? In S. Forrest (ed.), 5^{th} *ICGA*

Mitchell, T.M. (1982). Generalization as search. *Artificial Intelligence*, 18:203-226.

Petersen, C.C. (1967). Computational experience with variants of the balas algorithm applied to the selection of R & D projects. *Management Science*, 13:736-750.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, 1:81-106.

Reynolds, R.G. (1994). An introduction to cultural algorithms. In A.V. Sebald & L.J. Fogel (ed.), 3^{rd} *Conference on Evolutionary Programming*, World Scientific, 131-139.

Schaffer, J.D., Caruana, R.A., Eschelman, L. & Das, R. (1989). A study of control parameters affecting on-line performance of genetic algorithms for function optimization. In J. Schaffer (ed.), 3^{rd} *ICGA*, 51-60, Morgan Kaufmann.

Schaffer, J.D. & Morishima, A. (1987). An adaptive crossover distribution mechanism for genetic algorithms. In 2^{nd} *ICGA*, 36-40, Morgan Kaufmann.

Schoenauer, M. & Xanthakis, S. (1993). Constrained ga optimization. In Forrest S. (ed.), 4^{th} *ICGA*, 573-580, Morgan Kaufmann.

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models*. John Wiley & Sons, New-York.

Sebag, M. (1996). Delaying the choice of bias: a Disjunctive Version Space approach. In L. Saitta (ed.), 13^{th} *ICML*, Morgan Kaufmann.

Sebag, M. & Schoenauer, M. (1994). Controlling crossover through inductive learning. In H.P. Schwefel (ed.), 3^{rd} *PPSN*. Springer-Verlag, LNCS 866.

Spears, W.M. (1991). Adapting crossover in a genetic algorithm. In R.K. Belew & L.B. Booker (ed.), 4^{th} *ICGA*, Morgan Kaufmann.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In J.D. Schaffer (ed.), 3^{rd} *ICGA*, 2-9, Morgan Kauffman.

Whitley, D. (1991). Fundamental principles of deception in genetic search. In G.J.E. Rawlins (ed.), *Foundations of Genetic Algorithms*. Morgan Kaufmann.