An Induction-based Control for Genetic Algorithms (Extended Abstract)

Michele Sebag[1,3] and Marc Schoenauer[2,3] and Caroline Ravise[1,3]

[1] LMS, Ecole Polytechnique, 91128 Palaiseau, France
[2] CMAP, Ecole Polytechnique, 91128 Palaiseau, France
[3] LRI, Université de Paris-Sud, Bâtiment 490, F-91405 Orsay, France

Abstract. This paper presents an induction-based control of genetic algorithms:
1- examples of the behavior of the genetic operators (crossover and mutation) are gathered;
2- rules characterizing disruptive operators are induced from the gathered examples;
3- last, these rules are used to reject operators classified disruptive.
Evolution is thereby speeded up. Experimental results on the well-known Royal Road problem and on a GA-deceptive problem are presented.

1 Introduction

Genetic Algorithms (GAs) are widely known as powerful optimization algorithms [1]. As such, they have been applied in the Machine Learning community, mainly to build classifiers systems since the seminal work of Holland [3].

In contrast with using GAs to reach ML goals, we propose to use inductive learning to control and speed up GAs. The main drawback of GAs is their slowness. This slowness is partly due to the fact that most good solutions are *fleetingly discovered* [7] : genetic operators, i.e. crossover and mutation, stochastically discover promising individuals but make them disappear as well.

This paper describes an induction-based control of crossover and mutation, achieved in a 3-step process : (a) a set of examples about the behavior of operators is gathered, then (b) rules characterizing disruptive operators are induced, and last, (c) these rules are used during the next evolution steps to reject operators classified disruptive.

The reader is assumed familiar with the basic GA; we refer to [1] for the terms used throughout this paper. Section 2 briefly illustrates the difficulties encountered by GAs on two problems [6, 14]. Section 3 poses the problem of controlling a GA as an induction problem. Section 4 presents an experimental validation on the problems introduced in section 2. Section 5 discusses the induction-based approach with respect to some related works [4, 12].

2 Motivation

Let us introduce on two problems the main difficulties encountered by GAs.

The Royal Road problem. The Royal Road problem was conceived by Mitchell and Holland [6, 7]; this fitness landscape is supposedly easy for GAs because of (a) its schema structure and (b) the fact that high-order schemas are obtained by crossing over low-order schemas. However, it is *not* easy: low-order schemas must be discovered again and again before they combine.

GA-deceptive problems. The idea lying behind GA-deception, is that a GA can be misled by a particular fitness landscape [1]. If the average fitness of schema H_1 is greater than that of schema H_2, the schema theorem [2] states that H_1 will be oversampled in the genetic population compared to H_2. But if H_2 actually contains the optimum, the optimum will likely be missed. Moreover, as claimed by Whitley [14], all challenging problems are, at least partially, deceptive.

The desired control should both decrease the chances of disruptive evolution (in order to overcome fleeting discoveries) and the need for selection (in order to limit the misleading of GAs by deceptive problems).

3 Controlling a GA through Induction

Which Examples ? The only available information (when dealing with non-artificial problems) is provided by observing the course of evolution. This observation does tell how a given operator behaves on given individual(s): it behaves well if the best offspring is fitter than the best parent, badly if the best offspring is less fit than the (best) parent, otherwise it is inactive. Such behavioral examples permit an induction-based control of genetic search:

- Disruptive operators will likely be observed to behave *badly*;
- From examples of *bad* and *good* operators, rules can be induced by a classical inductive algorithm [5, 8].
- In further evolution steps, these rules allow to reject operators classified *bad*, thereby decreasing the chances of disruptive evolution.

This induction-based control thus speeds up evolution, by preventing it to go backward. Another possibility consists in rejecting operators classified *inactive*, thus preventing evolution to "get asleep". This latter heuristic was found very useful in the last stages of evolution: when population tends to be homogeneous, most operators are inactive; rejecting them brings a significant improvement [10].

How to Represent Examples ? In a binary frame, a crossover operator c can be represented by a bitstring $(c_1, ..c_N)$, termed *crossover mask* [13]. When crossing over individuals x and y, the first offspring inherits bit i from parent x iff $c_i = 1$, otherwise it inherits bit i from parent y (and symmetrically for the second offspring). Similarly, a mutation operator m can be represented by a mutation mask $(m_1, ..m_N)$: when mutating individual x, bit i is flipped iff $m_i = 1$.

An example is so composed of (a) the description of the operator and (b) the class of the operator, *bad, good* or *inactive* according to the effects of the operator w.r.t. the individual(s) it is experimented on. This representation leads to a control termed *general control.* Another possibility is to include the description of

the (best) parent the operator applies on, into the example description; this leads to more consistent examples since the behavior of an operator likely depends on the individual(s) it applies on. This latter possibility leads to another kind of control, termed *dedicated control*, which cannot be discussed here for the sake of brevity; more details can be found in [11].

Coupling GA and Induction. The coupling involves the following steps:
1. *Init.* During the first M generations, a classical genetic evolution takes place.
2. *Examples Gathering.* Let P be the size of the genetic population; either
2.1 P crossover examples are gathered, by randomly generating 2-point crossover masks and applying them on the current population; or
2.2 P mutation examples are gathered, by randomly generating mutation masks and applying them on the current population.
3. *Knowledge Building.* From these behavioral examples, rules are induced.
4. *Knowledge-guided Evolution.* In the next generations, crossovers or mutations classified *bad* according to the current rule-set, are rejected.
5. *Transition.* After M generations, the population has evolved and the rule-set may be no longer accurate. Go to step 2.

4 Experiments

We used a GA based on standards [1], with selection by roulette wheel with linear fitness scaling ; the selective pressure is set to values 1.2 and 2. Two-point crossovers are performed at a rate of 0.6. Mutation is performed at a rate of 0.05 per individual : in a mutated individual, every bit is flipped with probability 0.016. Evolution stops either after 1000 generations or when the fitness is constant over the population.
We used a star-like learner called *Constraint-Based Induction* [9]. Its complexity is $\mathcal{O}(N \times P^2)$, N being the size of the problem and P the number of examples.

Reference results are provided by running a standard GA (a) without control and (b) with a GA-based control, inspired from Spears [12]. We experimented both the crossover and the mutation control (results including simultaneous control of crossovers and mutations can be found in [11]). All results are averaged on 30 independent runs.

Parameters		Royal Road				Ugly Problem			
Sel. Press.	Pop. Size	No Control	Cross. Control GA	Cross. Control ML	Mut. Control	No Control	Cross. Control GA	Cross. Control ML	Mut. Control
1.2	100	10 (32)	43 (35)	20 (38)	57 (71)	10 (8)	7 (15)	3 (10)	53 (20)
2.0	100	7 (7)	10 (12)	3 (11)	17 (46)	20 (3)	17 (3)	17 (3)	53 (9)

Table 1. Percentage of success (nb of evaluations in thousands)

Experiments consider the Royal Road problem (2) and the Ugly problem [14] composed of 10 concatened elementary GA-deceptive problems. Table 1 shows the percentage of success (hit the optimum), together with the total number of

function evaluations needed to reach the optimum, for a population size 100. Partial conclusions are that: control is not useful when the population size is large; but control can significantly improves the results obtained with a small population. The control of mutation appears significantly more efficient than both kinds of crossover control, especially on the GA-deceptive problem.

5 Discussion

The crossover control has been addressed in two ways in the literature. Levenick [4] proposes to separate low-order schemas by zones of bits termed *introns*, not involved in the fitness computation. The disruptive effects of n-point crossovers can thereby be significantly limited [7]. However, such modifications of the problem representation require rather good insights in its solution (e.g. *a priori* knowledge of the position of the relevant low-order schemas) which greatly limits the method.

Spears [12] proposes to add an extra bit to the representation of individual; this extra bit rules out whether the individual is to be crossed according to a 2-point or a uniform crossover. Genetic search thus optimizes both the individual itself, and the kind of crossover most suited to this individual. Note however, that induction allows for a much more precise control: rules can tell *where* an operator should or should not intervene (depending on the individuals at hand in the case of dedicated control). Of course, a GA-based control could be similarly precise, but at the expense of doubling the size of the representation.

The cost of the induction-based control is decomposed in two parts. In terms of fitness calculations, it implies an overhead of $(M + 1)/M$, if learning is performed every M generations. Besides, it implies the learning and classification cost — that are polynomial and do not depend on the complexity of the fitness function. So, when dealing with expensive fitness functions, the extra-cost of an induction-based control should be negligible compared to the savings in terms of the number of fitness calculations needed to reach a good solution.

6 Conclusion

This paper has presented an induction-based control of GAs. On the ML side, this work is an application of induction; the difficulty consisted in posing the problem of controlling a GA as a machine learning problem.

On the GA side, it appears that induction offers powerful and flexible means to control a GA. Moreover, this approach gives unexpected insights into the roles respectively devolved to crossover and mutation along genetic search. The fact that mutation control is much more effective than crossover control is counterintuitive, since the crossover rate is much grater than the mutation rate. A tentative explanation is that nothing can counteract the disruptive effects of mutation, except control; in opposition, the sampling of the population can limit to a great extent, the disruptive effects of crossover (especially in the end of evolution). Much more experiments are needed, of course, to validate this hypothesis.

References

1. D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison Wesley, 1989.
2. J. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, 1975.
3. J. Holland. Escaping brittleness : The possibilities of general purpose learning algorithms applied to parallel rule-based systems. In R.S Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning : an artificial intelligence approach*, volume 2. Morgan Kaufmann, 1986.
4. 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 the* 4^{th} *International Conference on Genetic Algorithms.* Morgan Kaufmann, 1991.
5. 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.
6. 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/Bradford Books, 1993.
7. 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.* Morgan Kaufmann, 1993.
8. J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
9. M. Sebag. Using constraints to building version spaces. In De Raedt L. and Bergadano F., editors, *Proceedings of ECML-94, European Conference on Machine Learning*, pages 257–271. Springer Verlag, 1994.
10. 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.
11. M. Sebag and M. Schoenauer and C. Ravise. A Note on the Control of GAs by Induction. Internal Report, LMS, Ecole Polytechnique, january 1995.
12. 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.
13. G. Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*, pages 2–9, 1989.
14. D. Whitley. Fundamental principles of deception in genetic search. In R.K. Belew and L.B. Booker, editors, *Proceedings of the* 4^{th} *International Conference on Genetic Algorithms.* Morgan Kaufmann, 1991.