

Evolving Rules for Nonlinear Control

Jason Bobbin and Xin Yao
School of Computer Science, University College,
The University of New South Wales,
Australian Defense Force Academy,
Canberra, ACT 2600, Australia.
email: j-bobbin@adfa.edu.au

Abstract.

Computational intelligence techniques in control have facilitated the automatic generation of control strategies with little or no human input about the system. The present research examines the use of evolution in the generation of rule-based controllers for difficult nonlinear problems, where the derived controllers are in a compact, comprehensible form. The approach is able to evolve rule structures simultaneously with the parameters required to automatically find controllers for non-linear control problems. Evolution enables the use of a complex, information rich data structure which expresses the solution to the control problem with information about how the system is being controlled.

1 Introduction

Evolutionary algorithms have been successfully applied to many function optimisation problems, and increasingly they have been applied to learning tasks [1, 2]. One of the principal differences between learning and optimisation tasks are the way that solutions are represented by the algorithm. In function optimisation the solution is represented as either real valued genes in a chromosome or as binary strings. When applying evolutionary methods to other problem domains there are many representations possible including complete computer programs [2].

The principal issue considered in the current work is that of representation of the controller. We wish to discover a controller which can communicate information about how the system is being controlled to other controllers and is also sufficiently comprehensible to share the knowledge discovered by the evolution with human operators.

The choice of representation, and operators, clearly affects the efficiency of the search. It is also a method of encoding information about the problem domain into the algorithm, since the only solutions that the evolutionary search may find are those that can be represented by the genes it is using. There will also exist biases in the representation which affect the trajectory that the evolution will take to arrive at the solution. There has been a variety of representations used to evolve controllers, including binary strings [3, 4], weights in a neural network [5, 6, 7], mathematical functions in a genetic program [2, Page 289][8] and parameters for a model of the system being controlled [7].

A strength of the evolutionary process is the fact that it can operate on a diverse range of structures. A representation can be chosen which is appropriate for the intended use of the solution. In this paper we demonstrate an automatic controller which operates on comprehensible rules to find a reliable, robust controller. The algorithm discovers information about the problem and the chosen representation allows that information to be transferable to a knowledgeable operator, or to another automated

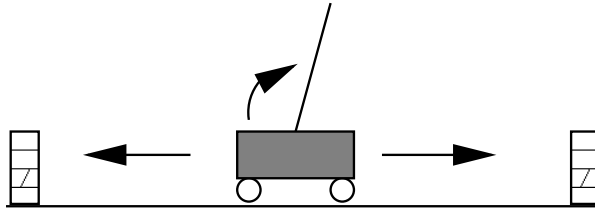


Figure 1: The cart-pole system

controller. Heuristic rules for how the problem is solved can be obtained directly by the algorithm as it discovers how to control the system.

The following section briefly describes the system being controlled and some previous research. Section 3 describes the technique used here in detail, and section 4 describes some results obtained with the method. Finally, Section 5 draws some conclusions.

2 Background

To demonstrate the evolution of a comprehensible rule-based controller we implement the representation and evolution algorithms on the well known cart-pole problem shown in figure 1. The task is to maintain the pole which is attached by a hinge to the top of the cart balanced for a set amount of time without ever pushing the cart over the end of the finite tracks (± 1 meter). The pole may only deviate from the vertical by a prescribed number of degrees (± 12 degrees). The task is complicated by only allowing a discrete force to be applied to the cart in either direction (± 10 Newtons). It is further complicated by starting the system from a randomly chosen position (± 0.5 m, ± 0.1 m/s, ± 2 deg, ± 1 deg/s). The problem allows us to examine the algorithms robustness in learning to control the system from a range of initial positions, and to control a non-linear system.

The equations and constants used to simulate the cart-pole system are given in Varšek et al. [4] and elsewhere.

The cart-pole system examined in this paper has been widely used as a test problem for different aspects of machine learning [3, 4, 5, 7, 9, 10]. Most research has concentrated on deriving a controller for the cart-pole system without a mathematical model of the system being incorporated into the learning algorithm. However, few studies have addressed the issue of how the learned knowledge can be transferred to solve similar problems and how the knowledge can be made more human comprehensible. Controllers which can communicate how they are deciding on the control to be applied require sophisticated languages in which to represent the controller. If we can provide evolution with a means of expressing the innovations which it discovers then the control problem can be solved and the controller understood.

In this paper an evolutionary algorithm operates on a data structure which is understandable by human operators, to provide a comprehensible solution to the cart-pole problem. By using evolution we are able to discover good rule structures and parameters to balance the pole, simultaneously. We address the issue of whether evolution can create comprehensible rule structures and parameters to balance the cart-pole system from a range of initial conditions. Previous approaches to this problem have mostly used human pre-defined partitions of the space which have then been optimized with control

values which balance the system, or have used connectionist approaches to discover neural networks which can balance the system, but cannot effectively communicate their solution to either human control experts or to other computer generated controllers.

3 Evolving Rule Based Controllers

Rule based controllers map regions of the state space to applicable actions. An evolutionary algorithm is proposed which can optimise the parameters associated with a rule set and thus find an appropriate discretization of the control system's state space automatically. A novel degenerate tree rule structure is proposed, which allows the evolutionary algorithm to explore the rule topology while simultaneously optimising the rule parameters. Thus a good discretization of the state space and rules to control the system can both be found without a-priori knowledge being given to the system about the optimal discretization or rule structure. Note that we do not assume symmetry in the parameter values.

3.1 Representing a Rule Set

Rule sets are evolved with an evolutionary algorithm operating on a novel degenerate tree rule representation. Each gene represents a simple rule which maps an area of the state space to an action. The genes are combined in a potentially complex topology which allows subsequent genes to modify the action of previously activated genes. The currently considered gene can have a number of exceptions, whose action will be executed if and only if the current gene is activated, and the exception genes rule is satisfied. The exception gene then becomes the current gene, and it too may have exceptions. The rule structure is a large tree-like structure, with each rule to the right refining the rules to the left. The rules are linked together in such a way that each member of the population which is evolved is a complete rule system which determines a complete solution to the problem. This degenerate tree rule structure is based on ripple down rules [11], which are noted as being very human comprehensible and succinct. Each rule has a context, and this is reflected in the way in which human experts think about rules.

At each node in the rule tree there is a comparison of the value of a state variable. If the comparison is true then the rules exception list is parsed. The exception list is identical in structure to the tree itself. If a node is not true then the 'if-not' list is parsed. Thus each node can have its area of influence refined by exception rules. If we consider the rule ($X_1, X_2, V_1, V_2 \in \mathbf{R}$)

$$\mathbf{IF } X_1 < X < X_2 \mathbf{ THEN action} = u_1$$

it defines an area of the state space based on the position (X) shown in the graph of position against velocity in figure 2. A mutation operator may then add an exception to the rule

$$\mathbf{EXCEPT IF } V_1 < V < V_2 \mathbf{ THEN action} = u_2$$

and we get the control policy shown in figure 2. The exception rule can have exception and if-not rules too, all of which refine the classification of the state space into control actions. In the chromosome in figure 2 the rules to the right of rule A are tested if and only if rule A is true. For rule D to be tested rule A must be true and rule B not true. For rule C to be tested rule A must be true and rule B. If rule C is then not found to be true, rule B would have it's proposed action implemented.

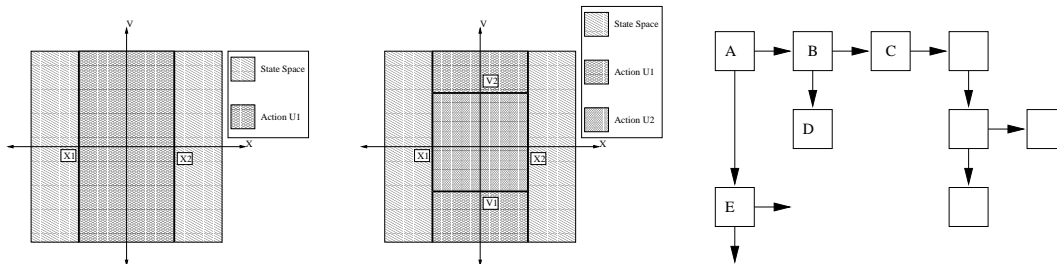


Figure 2: The area classified by a rule and a rule and exception rule, and the structure of the rule set

3.2 Mutation and Recombination of Rule Sets

The rule tree is modified by a number of operators. The most basic are the add and deletion operators, which add or remove a rule to the tree. There are also topological mutations. They modify the order in which rules are considered, and slightly change the resulting classification of the state space. Crossover swaps rules and their associated exception's between rule sets in the population. The probability of mutation and crossover occurring is decided by a self-adaptive scheme based on evolution strategies (ES) [12]. This allows the probability of structural mutations to change during the run on the basis of their past utility in generating fit offspring.

The parameter values associated with a solution are modified by another ES algorithm and passed on to the children of the solution. There is no crossover or swapping of parameter values between individuals in the population. This makes the parameter values analogous to a cultural behaviour which is passed on to the child by the parent. This cultural information instructs the child how to use the rule set it has been born with. When crossover swaps parts of another solutions rule structure, the way that the rule structure is used in the new solution is not the same as the previous solution since the parameter values for the new rule set will be different.

In each generation the probability of using a discrete mutation operator such as adding or deleting a rule is updated according to the ES stepsize formula [12, Page 144]. The new probability is coded onto the individual so that good rates of mutation and recombination can be applied at each stage in the search. The parameters are updated according to a standard ES implementation [12, Page 144].

When evaluating a controller the fitness associated with the controller is the average number of generations which the controller balanced the pole for from 5 random initial states. The selection is (μ, λ) , where $\frac{\mu}{\lambda} = \frac{1}{7}$. There are λ offspring born each generation from the best μ children of the previous generation. Using a non-elitist selection method prevents a controller receiving a 'lucky' set of random states and dominating the population without the need for re-evaluating controllers.

4 Experimental Studies

A number of runs were conducted using a population of 200 rule sets over 250 generations. The best controller from each run was able to balance the cart-pole system for a million time steps from the edges of the initial random region, and is considered to have solved the problem.

The output from a typical run is shown in figure 4. The changes in the probability of applying different operators at each generation is also shown. Note that the probability

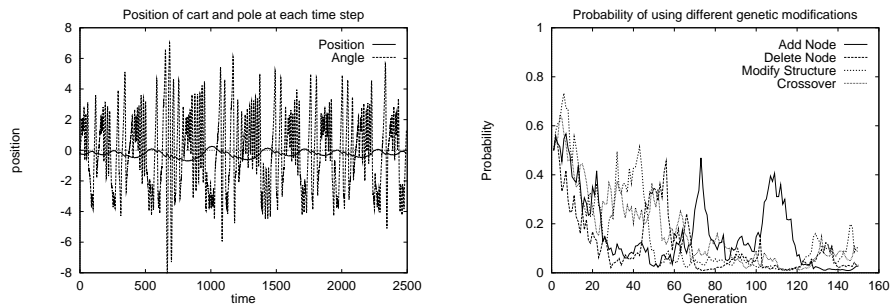


Figure 3: **Left:** The position of the cart in meters and the angle of the pole from vertical in degrees. **Right:** The probability of applying different operators at each generation.

of applying operators decreases as good structures are discovered. This decrease is entirely driven by selection, the update formula for the probabilities [12, Page 144] is neutral with respect to increasing or decreasing the probabilities over time. Selection does not monotonically decrease the probabilities because at certain times during the run some operators are of more value to the current structures compared to other times during the run.

4.1 Rule Sets Produced

A typical rule set found is shown in table 1 with the parameters in table 2. The rule set is very similar in nature to the human derived rules presented in [4]. The controller is principally concerned with the position of the pole, always moving in a direction to balance the pole when it is not balanced. When moving the cart, however, the controller moves to accelerate the cart to the left even while it is going left (Rule 2) so that the pole is unbalanced and the cart is moved back to the right as the pole becomes balanced.

The rule set shows how the representation communicates how the controller is controlling the cart. There are many ways this information can be used.

1. IF θ very negative then push LEFT
2. IF x' very negative then push LEFT EXCEPT IF θ' very positive then PUSH RIGHT
3. IF θ very positive the push RIGHT
4. IF θ' very negative then push LEFT
5. IF x very negative then push RIGHT EXCEPT IF θ is small then PUSH LEFT
6. IF θ is small then push RIGHT

Table 1: A typical rule set found for the cart-pole control problem

	Very Negative	Very Positive
x	-0.155 m	0.261 m
x'	-0.154 m/s	0.456 m/s
θ	-3.62 deg	1.99 deg
θ'	-8.27 deg/s	11.6 deg/s

Table 2: Parameters for a rule set to control the cart-pole problem

5 Conclusion

An evolutionary algorithm operating on a comprehensible rule set is outlined to solve control problems. The representation allows the knowledge discovered by the algorithm to be interpreted by human operators or transferred to other controllers. By utilizing an information rich data structure the evolutionary process can be exploited to provide insights into how difficult problems may best be solved. Evolutionary learning discovers a lot of information about the problem it is solving, however this information has not been available in any form other than the final point discovered. In control problems there is the potential for the evolution to learn to control the problem, and also to communicate *how* it has done so.

The form of rule sets considered in this paper enables classifications of the state space based on the values of the co-ordinate variables but not on combinations of co-ordinate variables. Future research will relax this condition and so enable more complicated control structures to be discovered.

References

- [1] D. Fogel, *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, 1995.
- [2] J. R. Koza, *Genetic Programming*. Stanford University. Cambridge, MA: MIT, 1992.
- [3] M. Odetayo and D. McGregor, "Genetic algorithms for inducing rules for a dynamic system," *International Conference on Genetic Algorithms*, 1989.
- [4] A. Varšek, T. Urbančič, and B. Filipič, "Genetic algorithms in controller design and tuning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, Sep/Oct 1993.
- [5] A. P. Wieland, "Evolving controls for unstable systems," *Connectionist Models: Proceedings of the 1990 Summer School*, 1991.
- [6] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Machine Learning*, vol. 22, pp. 11–33, 1996.
- [7] D. Fogel, "A "correction" to some cart-pole experiments," *Evolutionary Programming V*, pp. 67–71, 1996.
- [8] K. Chellapilla, "Automatic generation of nonlinear optimal control laws for broom balancing using evolution programming," *Proceedings of the 1998 IEEE Conference on Evolutionary Computation*, pp. 195–200, 1998.
- [9] C. X. Ling and R. Buchal, "Learning to control dynamic systems with automatic quantization," *Adaptive Behaviour*, vol. 3, no. 1, 1994.
- [10] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, And Cybernetics*, vol. 13, no. 3, 1983.
- [11] P. Compton and D. Richards, "Taking up the situated cognition challenge with ripple down rules," *International Journal of Human Computer Studies, Special Issue on Situated Cognition*, 1998.
- [12] H.-P. Schwefel, *Evolution and Optimum Seeking*. 605 Third Avenue, New York, NY 10158-0012, United States of America: John Wiley and Sons, 1994.