

Automatic Discovery of Relational Information in Comprehensible Control Rules by Evolutionary Algorithms

Jason Bobbin

Department of Soil and Water, University of Adelaide, Glen Osmond, South Australia 5064

Xin Yao

School of Computer Science, The University of Birmingham, Edgbaston, Birmingham B15 2TT, UK

Abstract

This paper examines the use of evolutionary methods in deriving comprehensible solutions in a relational problem domain. Evolution is used to manipulate a rule set with pre-defined relational functions to control the two-pole cart balancing problem. The evolved structures demonstrate that the representation can make use of relational information in a comprehensible rule set. Comprehensible solutions allow elucidation of the underlying behaviours and motivations of the control strategy derived, and also allow this knowledge to be shared and reused. This is demonstrated by solving the single pole balancing problem and using the rule set to seed solutions of the two-pole balancing problem.

1 Introduction

A principal aim of automated knowledge acquisition algorithms is to discover knowledge about the underlying problem domain. Discovered knowledge is only of use for communication and elucidation of processes if it is presented in a suitable representation or encoding. There has been a large array of representations used in evolutionary computation, from binary strings to lisp programs, and this ability to utilize different representations within a single computing paradigm allows a large range of problems to be addressed by evolutionary computation.

Solutions to many knowledge acquisition tasks depends on techniques capable of representing complex relationships between attributes of an input object. When the nature of these relationships are known before hand, or can be guessed, then there are clear advantages in having solutions representations which can explicitly search these relations. In this paper we show how evolution can be used to tune a simple relation and incorporate it in a controller for a difficult unstable system. The problem representation is based on a rule set, and is capable of communicating the knowledge which is automatically acquired by the evolutionary process. This is demonstrated by using the algorithm to solve a related, simpler problem and use that solution to provide better results in the more complex problem.

Self-adaptation is widely used in evolution strategies (ES) and evolutionary programming (EP), facilitating the evolution of real-valued and other representations. The use of self adaptation makes evolutionary search more efficient and allows the setting of parameters such as mutation and crossover rates to emerge from the evolutionary process. Evolutionary search is the product of a complex interaction between representation, operators, fitness function and selection methods. Self-adaptation removes some of the tuning parameters from the algorithm, making it easier to apply to new representations with new operators.

In this paper we present an evolutionary method which utilizes self-adaptation to construct comprehensible rule-set classifications of the state space of the control system, while simultaneously optimizing the parameters associated with the rule set.

The rest of this paper is organized as follows. Section 2 briefly describes the system being controlled and some previous research. Section 3 discusses the technique used here in detail, and section 4 presents some experimental results obtained with the method. Finally, Section 5 draws some conclusions.

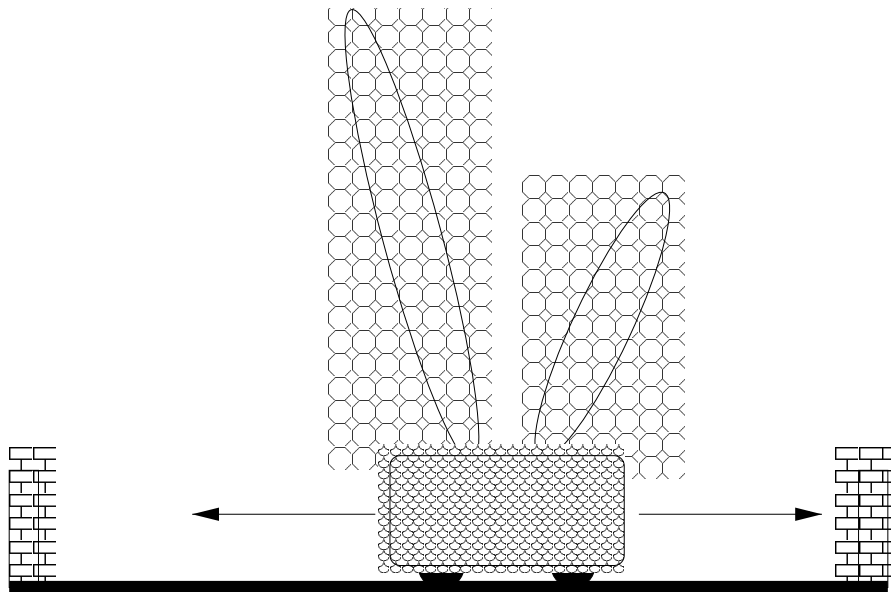


Figure 1: The two pole balancing problem

2 Background

The single pole balancing problem has been used extensively to test ideas in machine learning. This problem can be solved by a single neuron neural network, and as such is not a difficult control problem, although it is an unstable system which is representative of a large class of control problems [1]. The two-pole problem contains some of the more complex dynamics typical of difficult control problems, and this extension is considered in this paper to show that an evolutionary rule set can be used to automatically control a difficult control system.

The two-pole balancing is shown in figure 2. The problem is solvable under the condition that the two poles are not the same length. The closer the natural frequencies of the two poles (dependant on the length of the poles) to each other, the smaller the region of the state space from which the system can be controlled becomes. Two other pole balancing problems are regularly used as test cases for machine learning; the single jointed pole and a pole with time varying mass. The equations of motion for all these systems are detailed in Wieland [1] where it is also stated that the two pole balancing problem considered in this paper is the most difficult of the four pole balancing problems.

This system has been used to demonstrate the ability of evolved recurrent and evolved feed-forward neural networks to control a complex dynamic system [1, 2]. In both these cases the control value offered by the controller was allowed to range in the interval $[-10, 10]$ Newtons. In this paper we implement a more difficult bang-bang control system where the controller can only produce the maximal forces of $\{-10, 10\}$ Newtons. To demonstrate the learning of relational information it is sufficient to consider the cart on an unbounded track, and this condition was used to obtain the preliminary results in this paper. The poles must remain within 15 degrees of vertical, and their lengths were 0.5 and 0.05, although lengths of 0.5 and 0.25 were also successfully balanced.

In order to balance the two-pole system the cart must be moved so that the angle of deflection of the shorter pole becomes greater than the deflection of the longer pole, even when this means moving the cart so that the longer and shorter poles angle of deflections move further from upright. In order for a rule based controller to achieve this it must have some method of evaluating the difference between the angle of the two poles. A relation which depends upon the values of the angles must be determined in order to control the system.

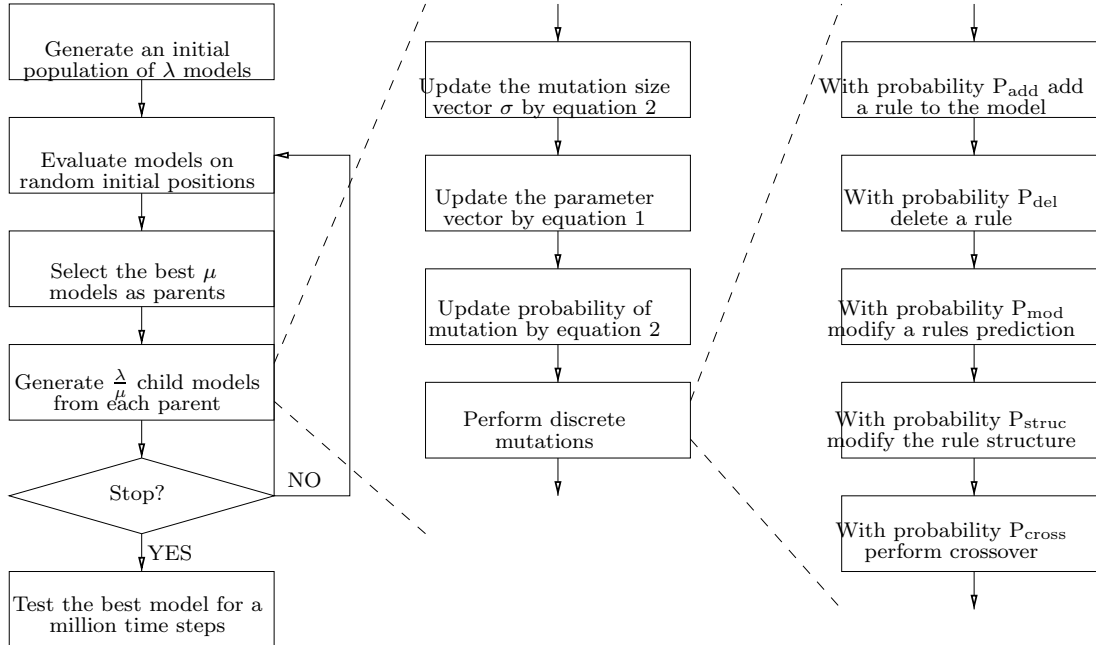


Figure 2: The main steps of the evolutionary process for discovering comprehensible and transferable rules (i.e., control models).

3 Evolving Control Rules

The main steps of our evolutionary algorithm are shown in figure 2. For the experiments presented in this paper the population size was set to 500 (i.e., $\lambda=500$) and the ratio of parent to children was set to approximately $\frac{1}{5}$ (i.e., $\mu = 100$). Each model is constructed from a rule set and a parameter vector. The rule structure is evolved simultaneously with the parameter vector. The parameter vector is evolved according to the paradigms of evolution strategies (ES) [3] and evolutionary programming (EP) [2] due to their superior performance in dealing with real numbers. Self-adaptation is used in our algorithm. The approach concentrates on using mutation and selection as the principal search operators although we do use crossover as a secondary search operator. The reason for emphasising mutation over crossover here is the same as the reason why mutation is more appropriate for evolving neural network structures and weights simultaneously [4, 5].

The selection scheme used is a truncation method referred to as (μ, λ) selection [3]. From a population of λ individuals the best μ are selected as the parents to the next generation. Each parent produces $\frac{\lambda}{\mu}$ children which form the next generation. This technique has no elitism, the best individual may be lost to the population. This method has empirically been found to promote successful self-adaptation [3]. More details about the steps in Figure 2 will be explained in the following subsections.

3.1 Representing a Rule Set

Rule-based controllers map regions of the state space to applicable actions. The evolutionary algorithm is able to optimise the parameters associated with a rule set and thus find an appropriate discretization of the control system’s state space automatically. Knowledge discovery is achieved through rule set creation. A novel degenerate tree rule structure is proposed, which allows the evolutionary algorithm to explore the rule topology while simultaneously optimizing 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. This is different from previous work in evolving control rules for the cart-pole problem. The rule structure used in our work is based on ripple-down rules which have been found to be highly succinct and comprehensible to human experts [6]. Figure 3 shows an example of the rule

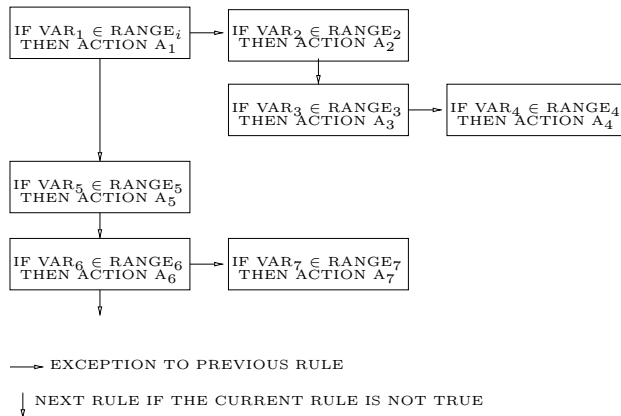


Figure 3: The rule structure.

structure.

In our 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 gene’s rule is satisfied. The exception gene then becomes the current gene, and it too may have exceptions. The rule topology is a tree-like structure, with each rule to the right refining the rules to the left. Each member of the population is a complete rule set.

Each rule has a context, and this is reflected in the way in which human experts think about rules. This is also important to the evolution, as it enables incremental changes to be made to the behaviour of the rule set by the addition of exceptions to more general rules. Maintaining behavioural links between parent and offspring is essential for successful evolution [4].

At each node in the rule tree there is a comparison of the value of a state variable. Each state variable is arbitrarily partitioned into three ranges, although this will be automatically done by a mutation operation in future work. The three ranges allowed for each variable is reduced to two parameters by considering the three ranges to be (MIN, p_1) , (p_1, p_2) and (p_2, MAX) , where MIN and MAX are smaller and larger than any possible value respectively (i.e., $-\infty$ and ∞ respectively). The parameter vector $(p_{11}, p_{12}, p_{21}, \dots, A_0, A_1, \dots)$ consists of the values

$$p_{jk} \quad \text{where } j \in \{1, \dots, \text{Inputs}\}, \\ \text{and } k \in \{1, 2\}$$

and the values A_l , $l \in [0, 1, 2]$, are either +1 or -1 corresponding to whether positive or negative force is to be applied to the cart, or 0 corresponding to an AND logical operation with any rule in the exception list. This is easily extended to other control values or functions for other problems.

A rule is a direct comparison between an input variable and a range of values. If the input variable’s value (VAR_i) falls within the range (RANGE_i) coded on the rule then the action (A_i) coded on the rule is performed. If a rule is true then its exception list is checked. If it is not then its ‘otherwise’ rule is checked. In this way the rule set is recursively parsed, with the last true rule having its prediction used as the output of the model on the given input example.

The evolutionary process is self-adaptive. There is a mutation vector σ which determines the size of the mutation step performed on a model, and which is co-evolved with the model. The parameter vector x is updated according to

$$\vec{x}' = \vec{x} \cdot \vec{N}(\vec{0}, \vec{\sigma}). \quad (1)$$

The strategy parameters, i.e., the mutation vector σ is updated according to

$$\sigma'_i = \sigma_i \cdot \exp(\tau' \cdot N(0, 1) + \tau \cdot N_i(0, 1)), \quad (2)$$

where $\tau \propto (\sqrt{2\sqrt{n}})^{-1}$ and $\tau' \propto (\sqrt{2n})^{-1}$. $N(0, 1)$ in equation 2 is a normally distributed random number with standard deviation one and mean zero. $N_i(0, 1)$ means the random variable is sampled anew for each value of i . σ_i denotes the i th component of $\vec{\sigma}$. The constant of proportionality for τ and τ' can be interpreted as a learning rate [7, page 72], and is set to 1. Recombination and covariance methods were not used in our self-adaptation scheme.

3.2 Mutation and Recombination of Rule Sets

The rule tree is modified by a number of operators. The most basic are the addition and deletion operators, which add or remove a rule to/from 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) [3]. 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 rule set are modified by another ES algorithm and passed on to the children of the individual. 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 solution's 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 step size formula given in equation 2 [3, 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 (equations 1 and 2) [3, Page 144].

When evaluating a controller the fitness associated with the controller is the average number of time steps which the controller balanced the pole for from 5 random initial states.

3.3 Representing Relational Information

As noted in section 2 controlling the two pole system relies on making control decisions based upon the difference between the angles of the two poles. To represent this we provide the algorithm with tunable linear functions which it uses in the same way as the other variables of the system. These functions have the form $f_1(\alpha) = \theta_1 + \beta\theta_2$ and $f_2(\beta) = \theta_2 + \beta\theta_1$ where $\alpha, \beta \in (-1, 1)$. The results of this function are treated as an attribute of the object being classified. The evolved decision is based on whether the tuned relation is in one of the ranges defined by the parameter vector. The parameters α, β are appended to the parameter vector and evolved by the self-adaptive scheme along with the rule set. The values that $f_1(\cdot), f_2(\cdot)$ can take are split into ranges with the terminal points of the ranges added to the parameter vector like the ranges of the attributes of the system.

4 Experimental Studies

A number of experiments have been conducted using a population of 500 individuals. A typical run of the algorithm on the two-pole problem produces the result shown in figure 4. The controller for this system is shown in figure 5. All the symbols are given numerical values by the parameter vector in the evolution, however using the symbolic values allows for easy comprehension. As an example, the algorithm defines a value less than -0.0479 radians to be a large negative difference in angles. The result shows how the relational information in the controller is being used. Negative values are to the left.

A second set of experiments was conducted where the shorter pole was held upright until the algorithm had successfully balanced the longer pole. The short pole was then allowed to move freely. The controller developed in balancing the one-pole problem is shown in figure 6. The final two-pole problem controller is shown in figure 7. The initial structures of the two rule sets shows how the system uses the knowledge it had gained in the one-pole problem for controlling the two-pole problem.

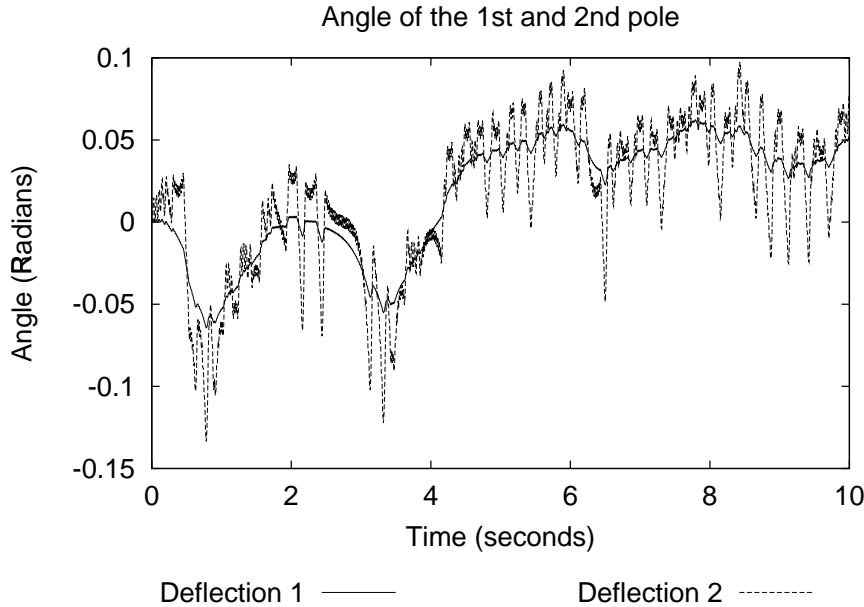


Figure 4: Deflection of the two poles

<p>IF $\theta_1 - \theta_2$ near zero AND θ'_2 is large then PUSH RIGHT IF θ'_2 large negative then PUSH LEFT IF $\theta_2 - \theta_1$ large then PUSH RIGHT IF θ'_1 is near zero then PUSH LEFT IF $\theta_2 - \theta_1$ near zero then PUSH RIGHT IF $\theta_2 - \theta_1$ large negative then PUSH LEFT</p>

Figure 5: Rule set for the evolved two-pole balancer

The evolved controllers rarely make use of rule exceptions in the two-pole problem, although they more frequently make use of them in the one-pole problem. Solving the one-pole problem first seems to result in a substantial decrease in the number of generations required to control the system, although more experiments are required to confirm this aspect.

5 Conclusion

There has been much work on evolving rules for nonlinear controllers. However, most of the work simply treats the evolutionary algorithm as a problem solver. This paper studies the issue of comprehensibility and transferability of evolved rules. The evolutionary algorithm is used as a problem solver as well as a heuristic and knowledge discoverer. The emphasis is on evolving rules which can be understood by human experts so that the knowledge and heuristics discovered can be applied to other problems, potentially without using the evolutionary approach.

In order to achieve our goal, a novel rule structure based on ripple-down rules has been adopted to represent the rule set. The ripple-down rules have been shown to be very human comprehensible and succinct [6]. They offer a very information-rich representation of rules. By utilizing an information rich data structure the evolutionary process can be exploited to provide insights into how difficult problems may best be solved. The simultaneous evolution of both rule structures and the parameters associated with them facilitates the discovery of novel rule sets. Self-adaptation used in our evolutionary algorithm enables the algorithm to find near optimal strategy parameters (i.e., various mutation and crossover probabilities) automatically.

The form of the rule sets adopted in this paper allows the use of information based on pre-defined

<p>IF θ'_1 large then PUSH RIGHT IF θ_1 is near zero then PUSH LEFT IF θ'_1 near zero then PUSH RIGHT IF cart velocity is large then PUSH LEFT</p>

Figure 6: Rule set evolved when one pole is held upright

<p>IF θ'_2 large then PUSH RIGHT IF $\theta_2 + \theta_1$ is near zero then PUSH LEFT IF θ'_1 large negative then PUSH RIGHT IF $\theta_2 + \theta_1$ is large AND θ'_2 is large negative then PUSH LEFT IF θ'_1 near zero then PUSH RIGHT IF θ'_1 large then PUSH RIGHT</p>

Figure 7: Rule set evolved after releasing the shorter pole θ_1 from being held upright

combinations of state space variables. The preliminary experiments presented demonstrate the ability of the evolutionary approach to tune and utilize these relationships, and to transfer knowledge between related problem domains. Future work will concentrate on quantifying the benefits of knowledge transfer and extending the classes of relationships which the algorithm can consider.

References

- [1] Alexis P. Wieland. Evolving controls for unstable systems. *Connectionist Models: Proceedings of the 1990 Summer School*, pages 91–102, 1991.
- [2] D.B. Fogel. *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, 1995.
- [3] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. John Wiley and Sons, 605 Third Avenue, New York, NY 10158-0012, United States of America, 1995.
- [4] X Yao. The importance of maintaining behavioural link between parents and offspring. In *Proc. of the 1997 IEEE Int'l Conf. on Evolutionary Computation (ICEC'97), Indianapolis, USA*, pages 629–633. IEEE Press, New York, NY 10017-2394, April 1997.
- [5] X Yao and Y Liu. Towards designing artificial neural networks by evolution. In *Applied Mathematics and Computation*, volume 91, pages 83–90, 1998.
- [6] Paul Compton and Debbie Richards. Taking up the situated cognition challenge with ripple down rules. *International Journal of Human Computer Studies, Special Issue on Situated Cognition*, pages 895–926, 1998.
- [7] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 198 Madison Avenue, New York, New York 10016, 1996.