Improving the Robustness and Encoding Complexity of Behavioural Clones

Rui Camacho[1,2] and Pavel Brazdil[1,3]

[1] LIACC, Rua do Campo Alegre, 823, 4150 Porto, Portugal
[2] FEUP, Rua Dr Roberto Frias, 4200-465 Porto, Portugal
rcamacho@fe.up.pt
http://www.ncc.up.pt/ tau
[3] FEP, Rua Dr Roberto Frias, 4200-464 Porto, Portugal
pbrazdil@ncc.up.pt
http://www.ncc.up.pt/ pbrazdil

Abstract. The aim of behavioural cloning is to synthesize artificial controllers that are robust and comprehensible to human understanding. To attain the two objectives we propose the use of the Incremental Correction model that is based on a closed-loop control strategy to model the reactive aspects of human control skills. We have investigated the use of three different representations to encode the artificial controllers: univariate decision trees as induced by C4.5; multivariate decision and regression trees as induced by CART and; clausal theories induced by an Inductive Logic Programming (ILP) system.
We obtained an increase in robustness and a lower complexity of the controllers when compared with results using other models. The controllers synthesized by CART revealed to be the most robust. The ILP system produced the simpler encodings.

Keywords: cognitive modeling, behavioural cloning, decision trees

1 Introduction

Two important issues in behavioural cloning concern the robustness and comprehensibility of the models constructed. For a successful application of the methodology it is required that the controllers constructed by the ML algorithm should be intelligible to human understanding and should replicate the robustness features of the human subject being modeled. It is further desirable that the model be constructed as automatically as possible.

An automatic process of reverse engineering human control skills offers a useful process of fast construction of controllers, specially in tasks where traditional Control Theory is not applicable. As pointed out by Sammut *et al.* (1992) it is also a very useful tool for training student pilots, particularly with regard to determining the aptitude of a candidate at an early stage of training. Hamm 92 ([5]) refers that Westland Helicopters Ltd uses helicopter engineering simulations, controlled by pilot models, for rotor loads and performance prediction studies. Because a human pilot is not included in the control loop, it is not

L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI 2167, pp. 37–48, 2001.

necessary for the helicopter simulation to run in real time — performance models may be run faster than real time for chart data production, and complex models may be run at less than real time, in cheap workstations. As Hamm points out, an accurate model of a human pilot is an important issue in this application. Urbančič *et al.*([18]) describe the application of the reverse engineering methodology to the control of a crane simulation in loading/unloading ships in a harbor.

Experiments reported in Sammut 92 and Michie and Camacho 94 revealed a lack in robustness and high complexity in the synthesised controllers. We made a first extension to the model used in those experiments introducing goals (*Goals model*) which allowed an increase in diversity in the datasets (the traces could be collected from different task plans). However, the robustness and comprehensibility objectives were not attained satisfactorily as shown in the Camacho 98 experiments. In Camacho 98 the IC model was introduced and the experimental results showed a significant improvement in robustness and complexity of the induced controllers. In those experiments C4.5 was used as the ML tool. C4.5 is not designed to take advantage of numerical classes and is restricted to univariate trees. In this paper we investigate how the previous results concerning robustness and complexity can be further improved using more powerful representation schemes. In the current study we compare the univariate trees results with multivariate decision and regression trees and also with clausal theories. Sammut *et al.* (1992) already suggested the use of First Order Logic to establish a set of operators with which the controllers should be composed. ILP systems offer a more natural way of providing background knowledge to the learning task than propositional learners. In the current study we profit also from the possibility of specifying constraints on the hypothesis language and therefore discarding "uninteresting" rules.

The experimental results indicate a significant improvement over the C4.5 results.

Although Reinforcement Learning algorithms and Neural Networks have achieved successful applications in control (see [21] and [10]) we did not consider in the current study approaches that, alone, do not produce symbolic and comprehensible descriptions. Comprehensibility is an important issue in applications we address. The use of such algorithms would require an approach similar to the one by Sammut (1988) where a decision tree learning algorithm was applied to extract the strategy produced by BOXES system (Michie and Chambers, 1968) after training to achieved to control the pole and cart. This line of research is currently pursued by Ryan and Reid ([11]) using hierarchical RL.

Application of predictive fuzzy control was already successfully done by Yasunobu and Hasegawa (1986) to the crane problem. However their approach is very time consuming and therefore there is place for the automatic synthesis of control rules from recorded performance of well trained operators. A classical

automatic controller by Sakawa and Shinido (1982) was reported to be vulnerable to wind and other unpredictable factors.

Suč and Bratko (1997) applied a novel approach whereby and ML algorithm induces an approximate model of the controlled system that enables the identification of subgoals the operator is said to pursue at various stages of the execution trace. They based their approach on the theory of Linear Quadratic regulator (LQ) controllers. The constructed clones are essentially LQ controllers. The controller is goal directed and takes into account the systems dynamics resulting in a high degree of robustness. The system is able to automatically identify the stages in a complete trace sequence.

Experimental results show that the LQ clones have no difficulty in controlling the rope swing doing better than regression tree clones of Urbančič and Bratko (1995).

The GRAIL methodology by Bain and Sammut (1999) was intended as an extension of the behavioural cloning methodology to handle goals and overcome robustness problems. It is in a prliminary stage but results show that it produces theories more compact than those of the Sammut *et al.* (1992) study and learns with much less traces. Bain and Sammut (1999) stress that the extension of GRAIL to use structured theories and background knowledge via first-order learning is most likely to further improve the performance of behavioural cloning.

The rest of the paper is structured as follows. In Section 2 we describe the methodology for reverse engineering human control skills together with some improvements on the original methodology. Section 3 discusses the models of Human Control Skills and describes our Incremental Correction model in detail. The experiments and the results obtained are reported in Section 4. In this section we present the comparative experimental results of using the different controller's encoding schemes. The last section presents the conclusions.

2 Reverse Engineering Human Control Skills

The methodology for reverse engineering human control skills, as described in [15], consists in the following six steps.

(1) Characterisation of the system being controlled as a set of state and control variables, representing the system status and decisions made by the human controller. (2) Definition of a task plan as a temporal sequence of stages. (3) Execution of the control task by the human controller according to the task plan. While performing the control task the system's state and control variables are recorded at regular intervals (the behavioural traces). (4) Pre-processing of the trace files to produce the ML tool datasets. (5) Induction of the decision trees using the ML tool. One tree for each control variable and for each stage. (6) Post-processing procedure by assembling all parts into an artificial controller. Apart from the induced code for determining the value

of each control variable, there is a hand-coded part that is responsible for switching the set of trees whenever there is a stage change in the task plan. The artificial controller replaces the human subject in the control cycle.

In the pre-processing phase (step 4) the data undergoes a series of filtering operations. The first filter splits the trace data into stages. A dataset is then created for each control variable and for each stage. Each dataset is then subject to another filtering operation that transforms samples into ML cases. In each sample, the value of the control variable is associated with the state variables and the other control variables of a sample recorded some time before. The time lag accounts for the human perceive-act delay. The control variable of the data set constitutes the class and the state, and the other control variables constitute the attributes. When the synthesised controller is run as an auto-pilot the perceive-act delay is introduced in the control cycle.

In the study of this paper we introduced two changes into the original methodology. First we require that in step 2 each stage corresponds to a *basic control manoeuvre* characterised by two kinds of goals discussed below. The second improvement includes the use of a *wrapper* ([6]) to facilitate the controller construction process by tuning the ML and model parameters (step 5).

The two kinds of goals, *Achievable goals* and *Homeostatic goals*, used here are imported from the work on AI planning by [4]. *Achievable goals* have a well-defined set of start and final states in the state space; arriving at anyone of the final state marks the achievement and termination of such a goal. These goals are the most common type in AI systems. *Homeostatic goals* are being achieved continuously. They do not terminate when the system is in one of the final states; when changes occur, and the state has changed, activity to re-achieve the final state is re-initiated. *Homeostatic goals* are like tracking certain required values.

A set of basic operations needs to be defined using the two kinds of goals defined above. During a basic operation the homeostatic and achievable goals remain constant. The task plan is then defined as a sequence of basic operations. In the flight domain, for example, the basic control operations are manoeuvres such as straight and level flight, levelled left turn, etc. In a levelled turn, for example, the homeostatic goals are the bank angle, the altitude and the climb rate and the achievable goal is the final heading. A flight plan is then a sequence of manoeuvres: straight climb, levelled left turn, levelled right turn, etc. We found that including more than a basic manoeuvre into a plan stage would increase the complexity of the control modules for that stage.

3 Models of Human Control Skills

The artificial controllers used in the experiments of [15], [8], [18] and [16] employ a two-level hierarchy of control: a high level module and a low-level one. The high-level module is intended to model the cognitive activity of human control skills. Such activity includes planning and monitoring. The low-level module in-

cludes the models for the reactive aspects of the human control skills. In the work of Stirling ([16]) the high-level module involves traditional knowledge acquisition (the influence matrix) whereas the low-level module was hand-coded using control theory PD controllers.

In [15], [8] and in the current study the high-level module is hand coded and its only role is to sequence the stages of the task plan. Here the high-level module is also responsible for establishing the context for the low-level module. It switches the low-level modules according to the stage of the task plan. The context is further specified by defining the goal values for the new stage. The low-level module has been implemented as a set of decision trees[1], one for each aircraft control and for each stage. At each stage only the trees constructed for that stage are active. This module is the only one induced from the behavioural traces and will be referred as the "model" from now on. We were only considering models for the reactive aspects of human control skills.

Sammut and Cribb (1990) and Whitley (1995) already pointed out the necessity of data diversity in order to produce robust controllers. However the model used in [15] and in [8] did not allow a great diversity of contexts in the behavioural traces. The *Goals model* (Camacho 98) improved matters allowing the traces to be collected under different task plans. Although there was an increase in robustness of the *Goals model* controllers the results were far from the behavioural cloning objectives. It was conjectured in [2] that the models in [15] and in [8] were not a plausible cognitive model. A controller induced within that framework acts as a mapping from a situation (or range of situations) to a control value. This model, therefore requires that the control values have to be memorised for each situation or range of situations. We conjecture that, for complex control tasks, humans do not use such strategy. All these supported the use of the Incremental Correction Model that we now describe.

3.1 The Incremental Correction Model

The Incremental Correction (IC) model is based on a closed-loop or feedback control strategy. There is a acceptable situation in which the controlled system is most of the time. The acceptable situation is characterised by a very small or non existing deviation from the homeostatic goal values. An acceptable situation requires no control change. Action is taken whenever the goal variables values deviate from the goal value. Usually a reasonable "wild guess" is used to make the first correction, specially if the deviation is large. After the first change, a sequence of three basic steps — (re)evaluation, corrections and waiting for the effects of the correction to become perceivable — takes place until the acceptability of the situation is restored. The amount of waiting time involved in real-time control is usually very small. The control strategy is adaptive, in the sense that the direction and magnitude of the correction is directly affected

[1] In some experiments we have used an ILP system (IndLog) that generated Prolog-like rules

by the previous change. If the previous change produces a too small reduction in the deviation then the direction of the next change is maintained and the magnitude of change increased. On the other hand, if the deviation is reduced too much or an overshooting is expected, then the direction is changed in the next cycle.

Human expert controllers would make a much more educated guess at the first corrective action than would a novice controller. An unexperienced controller would require a longer sequence of corrective actions than the expert. An expert may even anticipate the system response and dispense the feedback information.

The equations to compute the control values within the IC model are:

$$\Delta\ control = \begin{cases} 0, & situation\ ok \\ \mathbf{f}(state\ vars,\ goals,\ other\ controls), & situation\ not\ ok \end{cases}$$

The IC model implements the equation by means of two modules. One module (Coarse Decision) is used to determine if the situation requires a change in the controls. If a change in the controls is required (*situation not ok*), then another module computes the values of the increment/decrement of the control variables. This module is referred to as Refined Decision module. Whenever a change to a control is made the controller uses a waiting time for the effects of the change to become perceivable (feedback). The IC model inherits the goals and the perceive-act delay from the Goals model. Both modules are induced using ML methods from the behavioural traces of the human subject.

The IC model assumes that a required state can be achieved using corrective actions in the form of increments on the current position of the control value device. This avoids memorisation of the magnitude of the control value for each undesirable situations. In consequence, this model is not only easier to implement, but also provides a more plausible model of human control behaviour.

4 The Experiments

4.1 Experimental Settings

The control task chosen for the experimental evaluation of the models consists in the control of a simulation of an F-16 aircraft performing a levelled left turn. A levelled turn is a nontrivial manoeuvre requiring a close coordination among the controls. The experiments were carried out using ACM 2.4 public domain flight simulator running on a HP Apollo *Series* 735 workstation. The author played the role of the human pilot necessary for this study. A detailed description of the empirical evaluation of the models can be found in Camacho (2000).

The data used in the experiments are traces of 90 levelled left turn missions performed by the author. Aircraft state and control variables were sampled every 0.1 second. For the levelled left turn the achievable goal is the *final heading*. The homeostatic goals are the *bank angle*, the *altitude* and the *climb rate*. For each mission the achievable and homeostatic goals (except climb rate) were randomly

generated from a range of admissible values. In all missions and at all times the aircraft was kept within the security envelope (see below). The missions were flown by instruments only, the landscape was not rendered, reducing the possibility of the pilot to use features not measured by the aircraft instruments.

The 90 missions were split into two halves, one for constructing the model and the other to estimate the predictive accuracy of the constructed model. The 90 missions contain approximately 481 000 samples.

The attributes used in the construction of the controller's *ailerons* trees were: bank angle; bank angle derivative; bank angle acceleration; pitch and; pitch rate. The attributes used in the construction of the controller's *elevators* trees were: altitude deviation; climb rate; climb rate derivative; bank angle; bank angle derivative and climb rate acceleration.

The *wrapper* is a simple cycle that generates combinations of parameters values, synthesises the controller's components with such parameters values and estimates the performance of the controller with the constructed components. In the current study the wrapper was applied to the synthesis of each controller component (i.e. a decision tree) separately, estimating its performance on an independent test set (at induction time). Although we are including model parameters such as the perceive-act human delay in the *wrapper* we have no guarantee the component will behave well at deployment-time. At deployment-time there are interactions among controls that are not taken into account by the estimates on independent test sets when a single component is constructed at a time. To overcome such limitation of our current study we envisage to include in the *wrapper* the simulator where the controllers are evaluated. Doing that we may synthesise and test the deployment performance of the whole controller and generate the best combination of parameters for the full set of controller components.

4.2 Performance Evaluation

The following error measures give an indication on how close the model is to the original system, the human controller. The Error Rate (ER) and the Root Mean Square Error (RMSE) were measured on an independent test set and defined as

$$\mathbf{ER} = 100 \times \frac{\sum_{i=1}^{N} \begin{cases} 0 & if\ cls'_i = cls_i \\ 1 & otherwise \end{cases}}{N} \qquad \mathbf{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (cls'_i - cls_i)^2}{N}}$$

where cls_i is the actual class and cls'_i is the predicted value. Since the Coarse Decision module of the IC model outputs an action/no action decision, that is, a non numerical decision, only the Error Rate is used in this case.

Complexity of the induced trees was estimated using the tree size.

The *robustness* of the controllers is estimated by the number of successful missions within the total used. A mission is successful if there is no crash between the initial and final points.

The flight *smoothness* is evaluated using a deviation measure associated with each of the three homeostatic attributes. For each homeostatic attribute the Mean Absolute Deviation (MAD) is measured using the definition

$$MAD = \frac{\sum_{i=1}^{N} |att_i - goal|}{N}$$

Associated with each homeostatic goal there is a maximum acceptable deviation for that variable values. The frontier of the n-dimensional region containing the maximum acceptable values for the homeostatic variables is called the *security envelope*. All missions flown by the human subject that produce the behavioural traces were flown within the security envelope. The boundary for the altitude deviation is 100 ft, for the climb rate is 1000 ft/min and for the bank angle is 3°. The performance is evaluated by measuring the flight time percentage spent outside the envelope.

4.3 Evaluating the IC Model

For the IC model all samples are initially considered. The target definitions to learn within the IC model are: "change the control or not" and if the previous decision is favorable to change the control then the next decision is "what is the increment/decrement value to use". However the samples with no change in the control value largely outnumber the samples with "events" (the ratio is 36:1). The unbalance of such a dataset affects the results and needs to be corrected. The adopted procedure is as follows. In the first decision (action/no action) consider all the "events" as belonging to the same class (action) and all non-event samples as the other class (noaction). Build a data set with two classes equally represented by sampling from the noaction cases a number of cases equal to the action cases. The tree constructed with such dataset is then used to filter the "events" and produce the dataset for the Refined Decision module. In such a filtering procedure an event case is retained if it is predicted as "action" by the tree of the Coarse Decision module, otherwise is discarded. In the dataset of the Refined Decision module the increment of the control variable is restored as the class value.

Results with Univariate Trees

In the first set of experiments we used C4.5 as the ML algorithm. The induced controller successfully flew all the 90 missions. As the smoothness results, are concerned the MAD values (see Table 1) are very close to the human performance and even surpass the human performance in two of the measurements (climb rate and bank angle). There is also an improvement in Error rate (see Table 2) that is nearly half of the values obtained with previous models (*Goals model*). When comparing with previous models, the tree size is substantially smaller.

Table 1. Deployment-time performance. The results for altitude, climb rate and bank angle represent the average of the MAD values weighted with flight time.

Model	altitude (ft)		climb (ft/min)		bank angle (°)	
	train	test	train	test	train	test
IC (C4.5)	36	34	95	94	0.5	0.5
IC (CART)	15	14	105	104	0.4	0.4
Human	22	21	194	194	0.70	0.71

Table 2. Independent test set results for tree learners. *CD* stands for Coarse Decision module. *RD* stands for Refined Decision module.

tool	**module**	**ailerons**			**elevators**		
		tree size nodes	**Error Rate** (%)	**RMSE** x1e4	**tree size** nodes	**Error Rate** (%)	**RMSE** x1e3
C4.5	CD	**83**	**38.6**	-	**65**	**42.4**	-
	RD	663	-	**4.0**	**143**	-	**2.1**
CART	CD	**47**	**37.3**	-	**21**	**37.5**	-
	RD	75	-	**1.8**	**15**	-	**1.5**

Results with Multivariate Trees

The Refined Decision control has to predict a numerical quantity and therefore we expected that CART would do better than C4.5 on those datasets since CART induces regression trees that minimise the RMSE. There was in fact an improvement of the CART trees over the C4.5 ones. The CART results on RMSE are nearly half the C4.5 results (Table 1). The controllers constructed with CART have smaller tree size than the ones constructed with C4.5 (Table 2) (compare the tree size of the CART controller — 15 nodes — with the C4.5 tree for the elevators control — 143 nodes). However the complexity of the internal nodes in the CART trees are higher since CART trees may have linear combination of attributes in the internal nodes. The internal nodes of CART's trees are much more difficult to interpret than the simple $>$ and $\leq$ tests used in the C4.5 trees.

The results with both univariate and multivariate trees were confirmed by repeating the experiments using four extra partitions of the train/test set.

Comparison with Human Controller Performance

The results of Table 1 show that the IC model controller constructed with CART outperformed the human controller in all of the three deviation measures whereas the C4.5 surpasses the human subject in two (climb rate and bank angle). The climb rate MAD of the controller constructed with CART is 54% the MAD of the human subject. The bank angle MAD value is 57% of the human subject and the altitude MAD value is 71%. These results represent a significant evidence of the "clean-up effect" (Michie *et al.*, 1990).

Results with Clausal Theories

The third set of experiments aims at further reducing the complexity of the induced controllers. For that purpose we used and Inductive Logic Programming (ILP) system called IndLog (Camacho 2000) and encode the controller components as clausal theories.

The IndLog system was provided with background knowledge that included the $<$ and $>$ relations available to C4.5 and CART and also linear discriminants capable of producing equations similar to CART trees internal nodes. IndLog was further provided with multivariate linear regression equations to predict numerical control values in the Refined Decision module. We expected that this latter facility would increase controller robustness. A linear equation would allow the controller to extrapolate from the training conditions and produce control values for large deviations not seen during the training. An ILP system like IndLog is able to use almost any kind of bakground knowledge (statistical models, geometrical models etc..). Furthermore the user may easily specify restriction to the hypothesis language that avoid "uninteresting" rules to be produced. Both of these features revealed important in the current study.

Table 3. Independent test set results for ILP system. *CD* stands for Coarse Decision module. *RD* stands for Refined Decision module.

	ailerons			**elevators**		
module	**theory size** clauses	**Error Rate** (%)	**RMSE** x1e4	**theory size** clauses	**Error Rate** (%)	**RMSE** x1e3
CD	6	37.3	-	9	37.1	-
RD	5	-	1.6	4	-	1.4

The induction-time results are very close to the ones obtained with CART as can be seen in Table 3.

So far we haven't been able to successfully fly the 90 missions with the controller induced by the ILP system. The main reason is due to the linear equations that may very easily produce instability in the control since they may extrapolate to large control values. We are considering imposing a limit in maximum and minimum control values.

5 Conclusions

The use of goals and attributes that measure deviations from the defined goals brings a significant improvement in the models. It makes possible to use data from different task plans when constructing the controllers and it enables to use the same controller in different circumstances. Being able to increase the diversity of the training data is an important issue when trying to construct robust controllers.

The main contribution of the IC model is a substantial increase in robustness of the new controllers. The flight simulation performance values surpassed those of the human subject performance on the same missions (clean-up effect).

The trees constructed within the IC model exhibit a smaller size than the ones from previous experiments in reverse engineering human control skills. Intelligibility of the models is an essential point in the success criteria and the small tree sizes are a good step towards their comprehensibility.

Multivariate trees and clausal theories evaluated in the study revealed a significant improvement over the univariate trees. ILP system allowed the specification of background knowledge and language constraints that improves comphensibility of controllers.

As future work we intend to include the flight simulator in the *wrapper* cycle in order to account for control interactions at deployment-time.

Acknowledgments. Thanks are due to Universidade do Porto and JNICT for the financial support during the PhD. The authors thank the Programa PRAXIS and FEDER and the Programa de Financiamento Plurianual de Unidades de I&D da JNICT.

References

1. M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*. Oxford University Press, Oxford, U.K., 1999. (to appear).
2. R. Camacho. Inducing models of human control skills. In *Proceedings of the European Conference on Machine Learning – ECML-98*, Germany, April 1998.
3. R. Camacho. *Inducing Models of Human Control Skills using Machine Learning Algorithms*. PhD thesis, Universidade do Porto, July 2000.
4. A. A. Covrigaru and R. K. Lindsay. Deterministic autonomous systems. *AI Magazine*, 12(3):110–117, fall 1991.
5. J. C. Hamm. The use of pilot models in dynamic performance and rotor load prediction studies. In *Proceedings of the Eighteenth European Rotorcraft Forum*, pages 15–18, Avignon, France, September 1992. Association Aeronautique et Astronautique de France.
6. H. G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In W. W. Cohen and H. Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129, San Francisco, California, June 1994. Morgan Kaufmann.
7. D. Michie, M. Bain, and J. Hayes-Michie. Cognitive models from subcognitive skills. In M. G. J. McGhee and P. Mowforth, editors, *Knowledge-Based Systems for Industrial Control*, pages 71–99. Peter Peregrinus for IEE, London, UK, 1990.
8. D. Michie and R. Camacho. Building symbolic representations of intuitive real-time skills from performance data. In D. M. eds. K. Furukawa and S. Muggleton, editors, *Machine Intelligence 13*, pages 385–418. Oxford University Press, Oxford, United Kingdom, 1994.
9. D. Michie and R. A. Chambers. Boxes: an experiment in adaptive control. In *Machine Intelligence 2*, pages 137–152. Oliver and Boyd, Edinburgh, 1968. eds. Dale, E. and Michie, Donald.

10. J. Randlov and P. Alstrom. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the International Conference on Machine Learning – ICML-98*, pages 463–471, Madison, Wisconsin USA, July 1998.
11. M. Ryan and M. Reid. Learning to fly: An application of hierarchical reinforcement learning. In P. e. Langley, editor, *Proceedings of the Seventeenth International Machine Learning Conference, ICML-2000*, pages 807–814, San Francisco, CA., 2000. Morgan Kaufmann Publishers.
12. Y. Sakawa and Y. Shinido. Optimal control of container cranes. *Automatica*, 18:257–266, 1982.
13. C. Sammut. Experimental results from an evaluation of algorithms that learn to control dynamic systems. In *Proceedings of the Fifth International Workshop of Machine Learning 88*, pages 437–443, Ann Arbor, Univ of Michigan, June 1988. editor John Laird.
14. C. Sammut and J. Cribb. Is learning rate a good performance criterion for learning? In *Proceedings of the Seventh International Workshop of Machine Learning 90*, pages 170–178, Texas, June 1990.
15. C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In *Proceedings of the Ninth International Workshop of Machine Learning 92*, pages 385–393, Aberdeen, U.K., 1992.
16. D. Stirling. *CHURPs: Compressed Heuristic Reaction Planners*. PhD thesis, University of Sydney, 1995.
17. D. Šuc and I. Bratko. Skill reconstruction as induction of lq controllers with subgoals. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence – IJCAI-97*, volume 2, pages 914–920, Nagoya, Japan, 1997.
18. T. Urbančič and I. Bratko. Reconstructing human skill with machine learning. In *The Eleventh European Conference on Artificial Intelligence*, pages 498 – 502, Amsterdam, Netherlands, 1994. ed. A. Chon.
19. T. Urbančič and I. Bratko. Controlling container cranes: A case-study in reconstruction of human skill. In *The Second International Workshop on Artificial Intelligence Techniques– AIT95*, pages 113–126, Brno, Czech Republic, 1995.
20. D. Whitley. Genetic algorithms and neural networks. In *Genetic Algorithms in Engineering and Computer Science*, chapter 11. John Wiley & Sons Ltd, 1995. Eds. J. Periaux and G. Winter.
21. B. Widrow, D. E. Rumelhart, and M. A. Lehr. Neural networks: Applications in industry, business and science. *Communications of the ACM*, 37(3):93–105, 1994.
22. S. Yasunobu and T. Hasegawa. Evaluation of an automatic container crane operation system based on predictive fuzzy control. *Control-Theory and Advanced Technology*, 2:419–432, 1986.