Learning Classifier Systems: A Brief Introduction

Larry Bull

Faculty of Computing, Engineering & Mathematical Sciences
University of the West of England
Bristol BS16 1QY, U.K.
Larry.Bull@uwe.ac.uk

[Learning] Classifier systems are a kind of rule-based system with general mechanisms for processing rules in parallel, for adaptive generation of new rules, and for testing the effectiveness of existing rules. These mechanisms make possible performance and learning without the "brittleness" characteristic of most expert systems in AI.

Holland et al., *Induction*, 1986

1. Introduction

Machine learning is synonymous with advanced computing and a growing body of work exists on the use of such techniques to solve real-world problems [e.g., Tsoukalas & Uhrig, 1997]. The complex and/or ill-understood nature of many problem domains, such as data mining or process control, has led to the need for technologies which can adapt to the task they face. Learning Classifier Systems (LCS) [Holland, 1976] are a machine learning technique which combines reinforcement learning, evolutionary computing and other heuristics to produce adaptive systems. The subject of this book is the use of LCS for real-world applications.

Evolutionary computing techniques are search algorithms based on the mechanisms of natural selection and genetics. That is, they apply Darwin's principle of the survival of the fittest among computational structures with the stochastic processes of gene mutation, recombination, etc. Central to all evolutionary computing techniques is the idea of searching a problem space by evolving an initially random population of solutions such that better – or *fitter* – solutions are generated over time; the population of candidate solutions is seen to adapt to the problem. These techniques have been applied to a wide variety of domains such as optimization, design, classification, control and many others. A review of evolutionary computation is beyond the scope of this chapter, but a recent introduction can be found in [Eiben & Smith, 2003]. In LCS, the evolutionary computing technique usually works in conjunction with a reinforcement learning technique.

Reinforcement learning is learning through trial and error via the reception of a numerical reward. The learner attempts to map state and action combinations to their utility, with the aim of being able to maximize future reward. Reward is usually received after a number of actions have been taken by the learner; reward is typically

delayed. The approach is loosely analogous to what are known as secondary reinforcers in animal learning theory. These are stimuli which have become associated with something such as food or pain. Reinforcement learning has been applied to a wide variety of domains such as game playing, control, scheduling and many others. Again, a review of reinforcement learning is beyond the scope of this chapter and the reader is referred to [Sutton & Barto, 1998].

Learning Classifier Systems are rule-based systems, where the rules are usually in the traditional production system form of “IF state THEN action”. Evolutionary computing techniques and heuristics are used to search the space of possible rules, whilst reinforcement learning techniques are used to assign utility to existing rules, thereby guiding the search for better rules. The LCS formalism was introduced by John Holland [1976] and based around his more well-known invention – the Genetic Algorithm (GA)[Holland, 1975]. A few years later, in collaboration with Judith Reitman, he presented the first implementation of an LCS [Holland & Reitman, 1978]. Holland then revised the framework to define what would become the standard system [Holland, 1980; 1986]. However, Holland’s full system was somewhat complex and practical experience found it difficult to realize the envisaged behaviour/performance [e.g., Wilson & Goldberg, 1989]. As a consequence, Wilson presented the “zeroth-level” classifier system, ZCS [Wilson, 1994] which “keeps much of Holland’s original framework but simplifies it to increase understandability and performance” [ibid.]. Wilson then introduced a form of LCS which altered the way in which rule fitness is calculated – XCS [Wilson, 1995]. In the following sections, each of these LCS is described in more detail as they form the basis of the contributions to this volume. A brief overview of the rest of the volume then follows.

2. Holland’s LCS

Holland's Learning Classifier System receives a binary encoded input from its environment, placed on an internal working memory space - the blackboard-like message list (Figure 1). The system determines an appropriate response based on this input and performs the indicated action, usually altering the state of the environment. Desired behaviour is rewarded by providing a scalar reinforcement. Internally the system cycles through a sequence of performance, reinforcement and discovery on each discrete time-step.

The rule-base consists of a population of *N* condition-action rules or "classifiers". The rule condition and action are strings of characters from the ternary alphabet {0,1,#}. The # acts as a wildcard allowing generalisation such that the rule condition 1#1 matches both the input 111 and the input 101. The symbol also allows feature pass-through in the action such that, in responding to the input 101, the rule IF 1#1 THEN 0#0 would produce the action 000. Both components are initialised randomly. Also associated with each classifier is a fitness scalar to indicate the "usefulness" of a rule in receiving external reward. This differs from Holland's original implementation [Holland & Reitman, 1978], where rule fitness was essentially based on the accuracy of its ability to predict external reward (after [Samuel, 1959]).

On receipt of an input message, the rule-base is scanned and any rule whose condition matches the external message or any others on the message list, at each position becomes a member of the current "match set" [M]. A rule is selected from those rules comprising [M], through a bidding mechanism, to become the system's external action. The message list is cleared and the action string is posted to it ready for the next cycle. A number of other rules can then be selected through bidding to fill any remaining spaces on the internal message list. This selection is performed by a simple stochastic roulette wheel scheme. Rules' bids consist of two components, their fitness and their specificity, that is the proportion of non-# bits they contain. Further, a constant (here termed β) of "considerably" less than one is factored in, i.e., for a rule *C* in [M] at time *t*:

$$\text{Bid}(C,t) = \beta \cdot \text{specificity}(C) \cdot \text{fitness}(C,t)$$

Reinforcement consists of redistributing bids made between subsequently chosen rules. The bid of each winner at each time-step is placed in a "bucket". A record is kept of the previous winners and they each receive an equal share of the contents of the current bucket; fitness is shared amongst activated rules. If a reward is received from the environment then this is paid to the winning rule which produced the last output. Holland draws an economic analogy for his "bucket-brigade" algorithm, suggesting each rule is much like the middleman in a commercial chain; fitness is seen as capital.

The LCS employs a steady state GA operating over the whole rule-set at each instance. After some number of time-steps the GA uses roulette wheel selection to determine parent rules based on fitness. Offspring are produced via mutation and crossover in the usual way and replace existing rules, often chosen using roulette wheel selection based on the reciprocal of fitness.

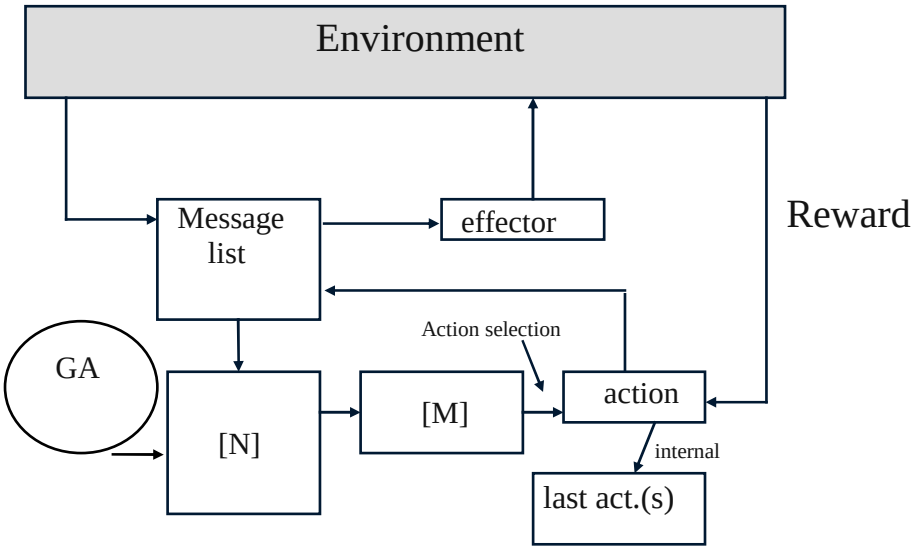

Fig. 1: Schematic of Holland's Learning Classifier System.

It is important to note that the role of the GA in LCS is to create a cooperative set of rules which together solve the task. That is, unlike a traditional optimisation scenario, the search is not for a single fittest rule but a number of different types of rule which together give appropriate behaviour. The rule-base of an LCS has been described as an evolving ecology of rules - "each individual rule evolves in the context of the external environment and the other rules in the classifier system." [Forrest & Miller, 1991].

A number of other mechanisms were proposed by Holland but for the sake of clarity they are not described here (see [Holland et al., 1986] for an overview).

Goldberg [1983] was the first to apply Holland's LCS to a real-world problem – gas pipeline control. Here the system received hourly readings from various sensors around a network, such as flow rates and pressures, and was required to deduce whether a leak was occurring and to set appropriate control of the pipeline inflow. Using a relatively small rule-base of 60 rules and a message list of size 5, Goldberg was able to train the LCS to become an effective controller after around 1000 days of simulated use. Other early applications of Holland's system include space vessel power management [Goodloe & Graves, 1988], sequence prediction [Riolo & Robertson, 1988], letter recognition [Frey & Slate, 1991] and modelling economic markets [Marimon et al., 1990]. The contribution to this volume by Tim Kovacs identifies many other applications of Holland's system.

3. Wilson's ZCS

As noted above, Wilson introduced the simple ZCS to increase understandability and performance. In particular, Wilson removed the message list and rule bidding (Figure 2). He introduced the use of action sets rather than individual rules, such that rules with the same action are treated together for both action selection and reinforcement. That is, once [M] has been formed a rule is picked as the output based purely on its fitness. All members of [M] that propose the same action as the selected rule then form an action set [A]. No wildcards are allowed in actions. An "implicit" bucket brigade [Goldberg, 1989] then redistributes payoff between subsequent action sets.

A fixed fraction - equivalent to Holland's bid constant - of the fitness of each member of [A] at each time-step is placed in a bucket. A record is kept of the previous action set $[A]_{-1}$ and if this is not empty then the members of this action set each receive an equal share of the contents of the current bucket, once this has been reduced by a pre-determined discount factor γ. If a reward is received from the environment then a fixed fraction of this value is distributed evenly amongst the members of [A] divided by their number. Finally, a tax is imposed on the members of [M] that do not belong to [A] on each time-step in order to encourage exploitation of the fitter classifiers. That is, all matching rules not in [A] have their fitnesses reduced by factor τ thereby reducing their chance of being selected on future cycles. Wilson notes that this is a change to Holland's formalism since specificity is not considered explicitly through bidding and the pay-back is discounted by $1-\gamma$ on each step (a

mechanism used in temporal difference learning to encourage solution brevity [e.g., Sutton & Barto 1998]). The effective update of action sets is thus:

$$\text{fitness}([A]) \leftarrow \text{fitness}([A]) + \beta\,[\,\text{Reward} + \gamma\,\text{fitness}([A]_{+1}) - \text{fitness}([A])\,]$$

ZCS employs two discovery mechanisms, a steady state GA and a covering operator. On each time-step there is a probability *p* of GA invocation. When called, the GA uses roulette wheel selection to determine two parent rules based on fitness. Two offspring are produced via mutation and crossover (single point). The parents then donate half of their fitnesses to their offspring who replace existing members of the population. The deleted rules are chosen using roulette wheel selection based on the reciprocal of fitness. The cover heuristic is used to produce a new rule with an appropriate condition to the current state and a random action when a match-set appears to contain less than averagely good rules, or when no rules match an input.

When ZCS was first presented, results from its use indicated it was capable of good, but not optimal, performance [Wilson, 1994][Cliff & Ross, 1995]. More recently, it has been shown that ZCS is capable of optimal performance, at least in a number of well-known test problems, but appears to be particularly sensitive to some of its parameters [Bull & Hurst, 2002]. Early applications of ZCS and its closely related forerunners BOOLE [Wilson, 1987] and NEWBOOLE [Bonelli et al., 1990] include medical data mining [Bonelli & Parodi, 1991] and modelling economic markets [e.g., Bull, 1999]. However, ZCS has been somewhat superseded by XCS.

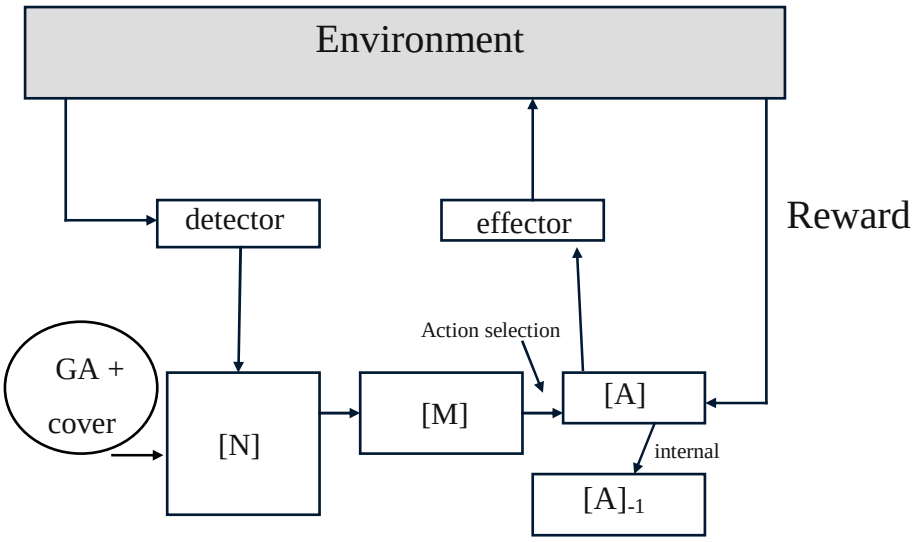

Fig. 2: Schematic of ZCS.

4. Wilson's XCS

The most significant difference between XCS (Figure 3) and most other classifier systems (e.g., ZCS) is that rule fitness for the GA is not based on payoff received (*P*) by rules but on the accuracy of predictions (*p*) in payoff. The intention is to form a complete and accurate mapping of the problem space (rather than simply focusing on the higher payoff niches in the environment) through efficient generalizations. In this way, XCS makes the connection between LCS and reinforcement learning clear and represents a way of using traditional reinforcement learning on complex problems where the number of possible state-action combinations is very large (other approaches have been suggested, such a neural networks – see [Sutton & Barto, 1998] for an overview).

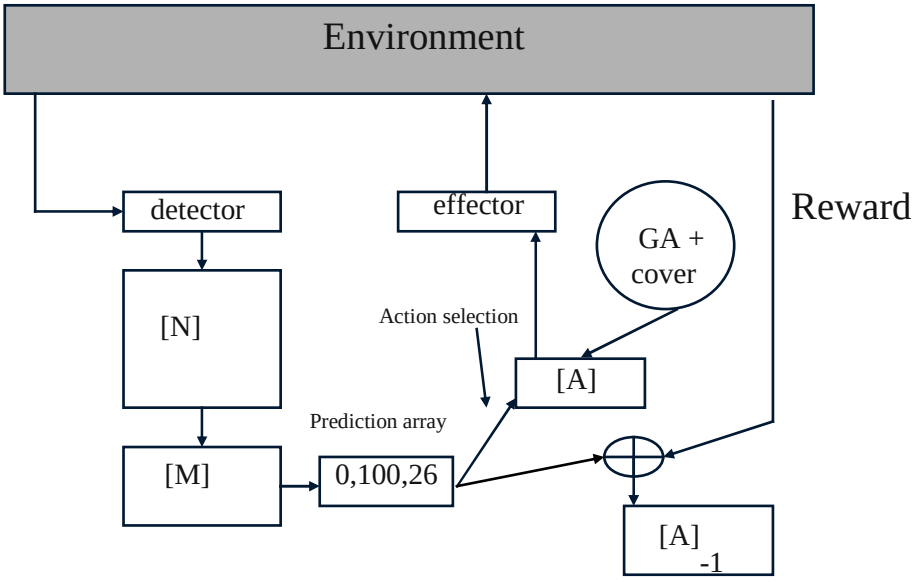

Fig. 3: Schematic of XCS.

On each time step a match set is created. A system prediction is then formed for each action in [M] according to a fitness-weighted average of the predictions of rules in each [A]. The system action is then selected either deterministically or randomly (usually 0.5 probability per trial). If [M] is empty covering is used.

Fitness reinforcement in XCS consists of updating three parameters, ε, *p* and *F* for each appropriate rule; the fitness is updated according to the relative accuracy of the rule within the set in five steps:

i) Each rule's error is updated: $\varepsilon_j = \varepsilon_j + \beta(| P - p_j | - \varepsilon_j)$

ii) Rule's predictions are then updated: $p_j = p_j + \beta(P-p_j)$

iii) Each rule's accuracy κ_j is determined: $\kappa = \alpha(\varepsilon_0/\varepsilon)^{\nu}$ or $\kappa=1$ where $\varepsilon < \varepsilon_0$

iv) A relative accuracy κ_j' is determined for each rule by dividing its accuracy by the total of the accuracies in the set.

v) The relative accuracy is then used to adjust the classifier's fitness F_j using the moyenne adaptive modifee (MAM) procedure: If the fitness has been adjusted $1/\beta$ times, $F_j = F_j + \beta(\kappa_j' - F_j)$. Otherwise F_j is set to the average of the current and previous values of κ_j'.

The maximum $P(a_i)$ of the system's prediction array is discounted by a factor γ and used to update rules from the previous time step. Thus XCS exploits a form of Q-learning [Watkins, 1989] in its reinforcement procedure, whereas Holland's system and ZCS both use a form of TD(0) (as noted in [Sutton & Barto, 1998]).

The GA acts in action sets [A], i.e., niches. Two rules are selected based on fitness from within the chosen [A]. Rule replacement is global and based on the estimated size of each action set a rule participates in with the aim of balancing resources across niches. The GA is triggered within a given action set based on the average time since the members of the niche last participated in a GA.

XCS is more complex than ZCS but results from its use in a number of areas have been impressive (e.g., see [Lanzi et al., 2000] for an overview). Indeed, the majority of contributions to this volume use XCS. For investigations of exactly how XCS works the reader is referred to [Butz et al., 2001]; a review of the known effects of its various mechanisms and current understanding of XCS is beyond the scope of this chapter. An algorithmic description of XCS can be found in [Butz & Wilson, 2001]. The first application of XCS to a real-world problem is described in the contribution to this volume led by Alwyn Barry. Others not included here include Ross et al. [2002] who have used it to solve constrained optimization problems and Danek and Smith [2002] who designed the layout of field programmable gate arrays.

5. Applications of Learning Classifier Systems: An Overview

This book, in keeping with the areas emerging from the field as amenable to LCS application, is divided into three main sections: data mining, modelling and optimization, and control. Each contribution describes the task to be solved, the form of LCS used and its (relative) performance.

5.1 Data Mining

Barry et al. – Data Mining using Learning Classifier Systems. LCS, and XCS in particular, show great promise in the area of data mining not least because they produce easily readable solutions in the form of unordered conjunctive logic rule sets. The authors show just how well LCS can do on a number of datasets.

Armano – NXCS Experts for Financial Time Series Forecasting. For some time now interest has been growing in the use of soft computing techniques to predict the stock exchange. This contribution presents results from using multiple XCS in conjunction with neural networks for predicting two financial markets.

Dixon et al. – Encouraging Compact Rulesets from XCS for Enhanced Data Mining. The use of machine learning techniques to explain aspects of the vast amounts of data now being gathered is rapidly increasing. This contribution describes findings from attempts to produce the readable solutions in their most compact form without sacrificing performance.

5.2 Modelling and Optimization

Smith – The Fighter Aircraft LCS. The agent paradigm [e.g., Kozierok & Maes, 1993] has opened new ways of using computer simulations to examine complex real-world systems. Machine learning is increasingly being used within the agents of such systems [e.g., Weiss & Sen, 1996]. In this contribution the use of LCS in the simulated testing of concept fighter aircraft is described, with known and new maneuvers being seen to emerge.

Hercog – Traffic Balance using Classifier Systems in an Agent-based Simulation. Here XCS is used within a multi-agent framework to model how best to re-arrange the capacity of different modes of transport typically found within a city. Further, the ability of the model to respond to changes to capacity is presented.

Bagnall – A Multi-Agent Model of the UK Market in Electricity Generation. In a similar vein to the previous contribution, this work describes the use of a multi-agent model to gain insights into the roles of aspects of, and the underlying dynamics of, the UK electricity supply market. A hierarchical inductive structure is used within each agent that includes two LCS.

Takadama – Exploring Organizational-Learning Orientated Classifier System in Real-World Problems. A novel form of LCS is described in this contribution which draws on ideas from management science to solve a scheduling task and another well-known constrained optimization task. The system, termed OCS, shows good performance and robustness to its parameters.

5.3 Control

Carse et al. - Distributed Routing in Communication Networks using the Temporal Fuzzy Classifier System – a Study on Evolutionary Multi-Agent Control. Fuzzy logic has long been used within the LCS framework due to its rule-based structure [e.g., Valenzuela-Rendon, 1989]. This contribution describes the use of a fuzzy-based LCS to control the routing nodes of a packet-switch network where each node is controlled by a separate LCS. The fuzzy system is hybridized with a well-known heuristic to give improved performance.

Browne – The Development of an Industrial Learning Classifier System for Data-Mining in a Steel Hot Strip Mill. Many industrial control problems are very noisy and dynamic making the effective use of traditional control methods difficult. This contribution describes how an LCS can be used to control such plants and how the logic produced can enable the engineers to learn more about the system.

Vargas et al. – Application of Learning Classifier Systems to the On-Line Reconfiguration of Electric Power Distribution Networks. Loss reduction in power distribution systems can be achieved by changing the status of distribution switches in order to uncover better paths of power flows. Larger gains can be achieved if on-line switching is allowed in order to consider varying location and time-dependent factors of distribution loads. This contribution explores the use of an LCS for the problem.

Bull et al. – Towards Distributed Adaptive Control for Road Traffic Junction Signals using Learning Classifier Systems. The effective control of large, complex distributed systems has to some extent eluded conventional techniques. This contribution details experiments with LCS to control road traffic junctions and compares performance with currently used algorithms.

6. Summary

Twenty five years after Holland and Reitman presented the first implementation of a Learning Classifier System, the ability of LCS to solve complex real-world problems is becoming clear, primarily due to the innovations introduced by Wilson. This article has given a brief introduction to LCS and the currently used versions of Holland's general formalism. The rest of the book brings together work by a number of individuals who have contributed to the understanding of how they can be used effectively. The final contribution by Tim Kovacs is a bibliography of such endeavours during the last twenty five years. Future work must apply LCS to a wide range of problems and identify characteristics which make the task suitable to solution with Learning Classifier Systems. Which form of LCS is most appropriate for which type of problem also needs to be established, along with continued refinement of the architectures and improved theoretical understanding.

Acknowledgements

Thanks to everyone involved in this edited collection: Professor Kacprzyk for agreeing to publish the book in his series, Professor Wilson for providing a Foreword, and, of course, the authors without whose efforts there would be no book. Thanks also to my fellow members of the Learning Classifier Systems Group at UWE for so many interesting discussions.

References

Bonelli, P. & Parodi, A. (1991) An Efficient Classifier System and its Experimental Comparison with two Representative Learning Methods on Three Medical Domains. In *Proceedings of the 4th International Conference on Genetic Algorithms*. Morgan Kauffman, pp. 288-295.

Bonelli, P., Parodi, A., Sen, S. & Wilson, S.W. (1990) NEWBOOLE: A Fast GBML System. In *International Conference on Machine Learning*. Morgan Kaufmann, pp. 153-159.

Bull, L. (1999) On using ZCS in a Continuous Double-Auction Market. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference – Gecco 1999*. Morgan Kaufmann, pp83-90.

Bull, L. & Hurst, J. (2002) ZCS Redux. *Evolutionary Computation* 10(2): 185-205.

Butz, M., Kovacs, T., Lanzi, P.L. & Wilson, S.W. (2001) How XCS Evolves Accurate Classifiers. In *Proceedings of the 2001 Genetic and Evolutionary Computation Conference – Gecco 2001*. Morgan Kaufmann, pp927-934.

Butz, M. & Wilson, S.W. (2001) An Algorithmic Description of XCS. *Advances in Learning Classifier Systems: Proceedings of the Third International Conference – IWLCS2000*. Springer, pp253-272.

Danek, M. & Smith, R.E. (2002) XCS Applied to Mapping FPGA Architectures. In *Proceedings of the 2002 Genetic and Evolutionary Computation Conference – Gecco 2002*. Morgan Kaufmann, pp912-919.

Eiben, A. & Smith, J. (2003) *Introduction to Evolutionary Computing*. Springer.
Forrest, S. & Miller, J. (1991) Emergent Behaviour in Classifier Systems. *Physica D* 42: 213-217.

Frey, P.W. & Slate, D.J. (1991) Letter Recognition using Holland-style Adaptive Classifiers. *Machine Learning* 6: 161-182.

Goldberg, D.E. (1983) Computer-aided Gas Pipeline Operation using Genetic Algorithms and Rule-learning. Ph.D. Thesis, University of Michigan.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Goodloe, M. & Graves, S.J. (1988) Improving Performance of an Electric Power Expert System with Genetic Algorithms. In *Proceedings of the 1st International Conference on the Applications of Artificial Intelligence and Expert Systems*. IEA/AIE-88, pp298-305.

Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Holland, J.H. (1976) Adaptation. In R. Rosen & F.M. Snell (eds) *Progress in Theoretical Biology*, 4. Plenum.

Holland, J. H. (1980) Adaptive Algorithms for Discovering and using General Patterns in Growing Knowledge Bases. *International Journal of Policy Analysis and Information Systems* 4(3): 245-268.

Holland, J. H. (1986). Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds) *Machine learning, an artificial intelligence approach*. Morgan Kaufmann.

Holland, J.H. & Reitman, J.H. (1978) Cognitive Systems Based in Adaptive Algorithms. In D. Waterman & F. Hayes-Roth (eds*) Pattern-directed Inference Systems.* Academic Press.

Holland JH, Holyoak KJ, Nisbett RE and Thagard PR (1986) Induction: Processes of Inference, Learning and Discovery, MIT Press.

Kozierok, R. & Maes, P. (1993) A Learning Interface Agent for Scheduling Meetings. In *Proceedings of the ACM SIGCHI International Workshop on Intelligent User Interfaces*. ACM Press, pp81-88.

Lanzi, P-L, Stolzmann, W. & Wilson, S.W. (2000) *Learning Classifier Systems: From Foundations to Applications*. Springer.

Marimon, R., McGrattan, E. & Sargent, .J. (1990) Money as a Medium of Exchange in an Economy with Artificially Intelligent Agents. *Journal of Economic Dynamics and Contro*l 14: 329-373.

Robertson, G. & Riolo, R. (1988) A Tale of Two Classifier Systems. *Machine Learning* 3: 139-159.

Ross, P., Schulenburg, S., Marin-Blazquet, J. & Hart, E. (2002) Hyper-heuristics: Learning to Combine Simple Heuristics in Bin-packing Problems. In *Proceedings of the 2002 Genetic and Evolutionary Computation Conference – Gecco 2002*. Morgan Kaufmann, pp942-948.

Samuel, A.L. (1959) Some Studies in Machine Learning using the Game of Checkers. *IBM Journal on Research and Development* 3: 211-229.

Sutton, R. & Barto, R. (1998) *Reinforcement Learning*. MIT Press.

Tsoukalas, L.H & Uhrig, R.E. (1997) *Fuzzy and Neural Approaches in Engineering*. Wiley.

Valenzuela-Rendon, M. (1991) The Fuzzy Classifier System: a Classifier System for Continuously Varying Variables. In *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann, pp346-353.

Watkins, C.J. (1989) Learning from Delayed Rewards. Ph.D. Thesis, Cambridge University.

Weiss, G. & Sen, S. (1996)(eds) *Adaptation and Learning in Multi-Agent Systems*. Springer.

Wilson, S.W. (1987) Classifier Systems and the Animat Problem. *Machine Learning* 2: 219-228.

Wilson, S.W. (1994) ZCS: A Zeroth-level Classifier System. *Evolutionary Computation* 2(1):1-18.

Wilson, S.W. (1995) Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2): 149-76.

Wilson S.W. & Goldberg D.E. (1989) A critical review of classifier systems. In *Proceedings of the 3rd International Conference on Genetic Algorithms*. Morgan Kauffman, pp. 244-55.