Foundations of Learning Classifier Systems: An Introduction

Larry Bull[1] & Tim Kovacs[2]

[1]School of Computer Science
University of the West of England
Bristol BS16 1QY, U.K.
`Larry.Bull@uwe.ac.uk`

[2]Department of Computer Science
University of Bristol
Bristol BS8 1UB, U.K.
`Kovacs@cs.bris.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

Learning Classifier Systems (LCS) [Holland, 1976] are a machine learning technique which combines evolutionary computing, reinforcement learning, supervised learning or unsupervised learning, and heuristics to produce adaptive systems. They are rule-based systems, where the rules are usually in the traditional production system form of "IF state THEN action". An evolutionary algorithm and heuristics are used to search the space of possible rules, whilst a credit assignment algorithm is 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; 1986a]. 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] and interest waned. Some years later, 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]. The following decade has seen resurgence in the use of LCS as XCS in particular has been found able to solve a number of well-known problems optimally. Perhaps more importantly, XCS has also begun to be applied to a number of hard real-world problems such as data mining, simulation modeling, robotics, and adaptive control (see [Bull, 2004] for an overview) and where excellent performance has often been achieved. Further, given their rule-based nature, users are often able to learn about their problem domain through inspection of the produced solutions, this being particularly useful in areas such as data mining or safety-critical control for example. However their combination of two machine learning techniques and potentially many heuristics means that formal understanding of LCS is non-trivial. That is, current formal understanding of, for example, Genetic Algorithms and Reinforcement Learning is significant but understanding of how the two interact within Learning Classifier Systems is severely lacking. The purpose of this volume is to bring together current work aimed at understanding LCS in the hope that it will serve as a catalyst to a concerted effort to produce such understanding.

The rest of this contribution is arranged as follows: Firstly, the main forms of LCS are described in some detail. A number of historical studies are then reviewed before an overview of the rest of the volume is presented. See [Barry, 2000] for more on early LCS.

2. Holland’s LCS

Holland's Learning Classifier System [Holland, 1986] 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*:

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

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 winners on the previous time step 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 (BBA), suggesting each rule is much like the middleman in a commercial chain; fitness is seen as capital. The reader is referred to [Sutton & Barto, 1998] for an introduction to reinforcement learning.

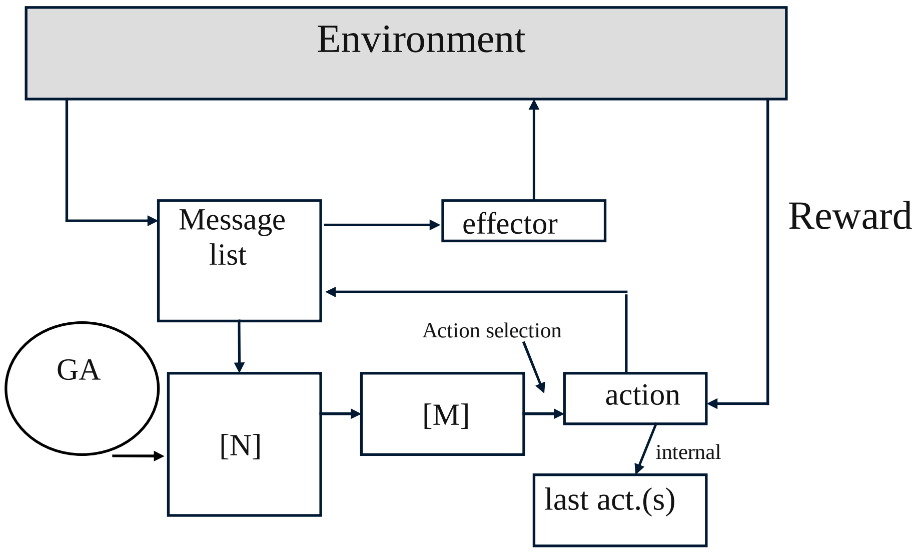

Fig. 1: Schematic of Holland’s Learning Classifier System.

The LCS employs a steady-state Genetic Algorithm operating over the whole rule-set at each instance. After some number of time-steps the GA uses roulette wheel selection to determine two parent rules based on their fitness relative to the total fitness of the population:

$$\text{Probabilty\_Selection}(\ C,t\ ) = \text{fitness}(\ C,t\ ) \ /\ \Sigma\ \text{fitnesses}(\ t\ )$$

The effect of this scheme is to bias reproduction towards those rules which appear to lead to higher reward from the environment. Copies are made of the chosen rules which are then subjected to two genetic operators: mutation and crossover. Mutation is applied probabilistically at a per-locus rate (e.g., 1/100) along the length of the rule and upon satisfaction the value at that locus is altered – typically, a locus becomes one of the other two possible values with equal probability. For example, if the above mentioned rule 1#1:0#0 experiences a mutation event on its last locus it could become 1#1:0#1 or 1#1:0##. Crossover begins by randomly choosing a position within the rules and then swaps them from that point to their end. For example, the two rules 000:000 and 111:111 which experience crossover at position two would become 001:111 and 110:000 respectively. The purpose of the genetic operators is to introduce new rules into the population based on known good rules with the aim of discovering better rules. The new rules then replace two existing rules, often chosen using roulette wheel selection based on the reciprocal of fitness. The reader is referred to [Eiben & Smith, 2004] for a recent introduction to evolutionary computing.

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).

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) and did not allow wildcards in actions. 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]. An "implicit" bucket brigade [Goldberg, 1989] then redistributes payoff in the subsequent action set.

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 (a mechanism used in temporal difference learning to encourage solution brevity [e.g., Sutton & Barto, 1998]). 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 considered this technique provisional and suggested there were better approaches to controlling exploration. The effective update of action sets is thus:

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

where $0 \leq \beta \leq 1$ is a learning rate constant. Wilson noted that this is a change to Holland's formalism since specificity is not considered explicitly through bidding and pay-back is discounted by 1-γ on each step. 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. The parents donate half their fitness 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 low quality rules, or when no rules match an input.

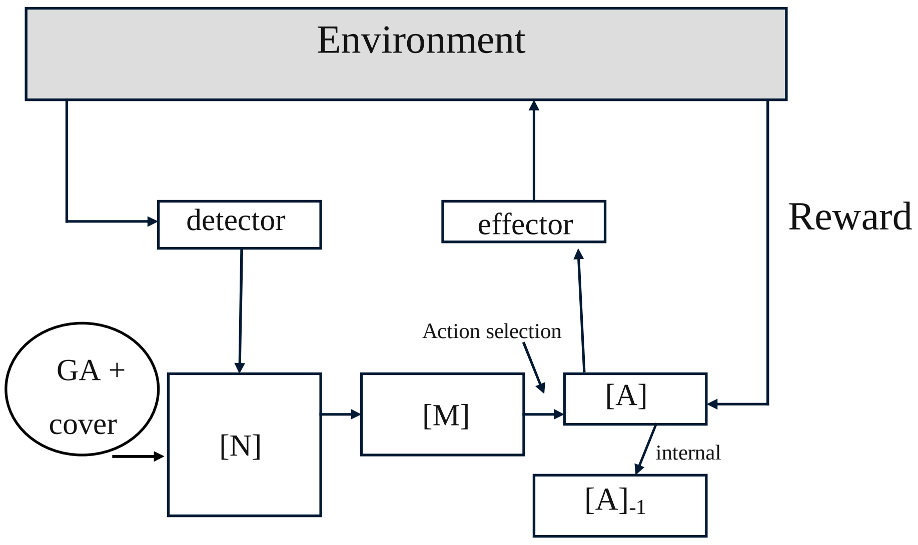

Fig. 2: Schematic of ZCS.

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]. It should be noted that ZCS has two closely related forerunners, namely BOOLE [Wilson, 1987] and NEWBOOLE [Bonelli et al., 1990].

4. Wilson's XCS

The most significant difference between XCS (Figure 3) and most other LCS (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*) of payoff. Hence, XCS has been termed an accuracy-based LCS, in contrast to earlier systems which were for the most part strength-based (also called payoff-based systems). The intention in XCS 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 RL terms, XCS learns a value function over the complete state/action space. In this way, XCS makes the connection between LCS and reinforcement learning clear and represents a way of using traditional RL 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).

XCS shares many features with ZCS, and inherited its niche GA, deletion scheme and an interest in accuracy from Booker's GOFER-1 [Booker, 1982].

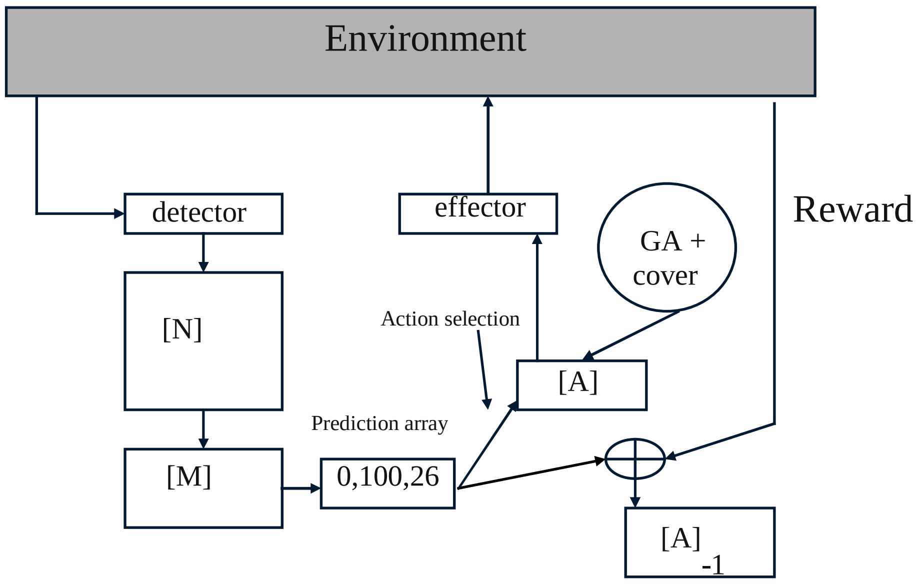

Fig. 3: Schematic of XCS.

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: $\varepsilon_j = \varepsilon_j + \beta(\mid P - p_j \mid - \varepsilon_j)$ where as in ZCS $0 \leq \beta \leq 1$ is a learning rate constant.

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

iii) Each rule's accuracy κ_j is determined:
$\kappa_j = \alpha(\varepsilon_0/\varepsilon)^\nu$ or κ=1 where $\varepsilon < \varepsilon_0$
where ν, α and ε_0 are constants controlling the shape of the accuracy function.

iv) A relative accuracy κ_j' is determined for each rule by dividing its accuracy by the total of the accuracies in the action 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 values of κ' seen so far.

In short, in XCS fitness is an inverse function of the error in reward prediction, with errors below ε_0 not reducing fitness. 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 1986 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 (after [Booker, 1989]).

XCS is more complex than ZCS but results from its use in a number of areas have been impressive. Wilson originally demonstrated results on the Boolean multiplexer function and a maze problem [Wilson, 1995]. Early on Kovacs emphasised its ability to learn complete, accurate, and minimal representations of Boolean functions [Kovacs, 1997]. XCS has since shown good performance on data mining tasks [e.g., Bernado, Llora & Garrell, 2002] and has been widely adopted in the LCS community; the majority of contributions to a recent volume on applications of LCS [Bull, 2004] used XCS. An algorithmic description of XCS can be found in [Butz & Wilson, 2001], while further details of XCS and an example execution cycle can be found in [Kovacs, 2004]. A brief overview of selected theoretical works now follows. We concentrate on pre-ZCS and XCS systems in order to complement the remaining chapters of this text, and on formal studies rather than experimental ones.

5. Previous Research on the Foundations of LCS

Since Learning Classifier Systems combine two machine learning algorithms, previous studies of their behaviour from a theoretical standpoint have tended to focus on one aspect over the other. The following historical review is divided to reflect this. Further related material is available in [Wilson & Goldberg, 1989; Lanzi & Riolo, 2001].

5.1 Rule Discovery: Evolutionary Algorithms

The term Evolutionary Algorithm denotes a family of stochastic problem solvers based on a population of solutions being manipulated by the neo-Darwinian processes of selection, recombination and mutation. The Genetic Algorithm, as briefly described above, is the most commonly used approach but recent work has included parameter self-adaptation [e.g., Bull et al., 2000] normally associated with Evolution Strategies [Rechenberg, 1973] and the later forms of Evolutionary Programming [e.g., Fogel, 1992], and the use of LISP S-expressions to represent rules [e.g., Lanzi, 1999b] as found in Genetic Programming [Koza, 1992]. Until the early 1990's, Holland's Schema Theorem [Holland, 1975] was the most widely used theoretical tool for understanding GAs and thus it was also used as a basis for some of the earliest work on Learning Classifier Systems.

Smith and Valenzuela-Rendon [1989] presented a simple proportion vector form of the canonical GA through which they considered the propagation of the set of eighteen rules with two-bit conditions and one-bit actions where there was no pass-through in the latter, i.e., the rules 00:0 to ##:1. Roulette wheel selection and single-point crossover were included in this infinite population generational model - a model based on the traditional scenario of replacing the whole population per reproduction cycle. The LCS was assigned a stimulus-response task, that is, a task under which each response from the LCS is rewarded immediately by the environment; three Boolean functions of varying difficulty were used. Initial results showed how a standard GA is unable to converge to a solution containing a full set of rules required to solve the given tasks. That is, the GA operated as it does in the standard function optimization scenario and simply sought solutions/rules which typically led to the highest fitness/reward only. They then examined the effects of fitness sharing in their model. Fitness sharing was highlighted by Booker [1982] as a way to prevent the GA population from clustering around such solutions. Simply, individuals are said to share the reward received with those who are similar to them in some way. In GA function optimization similarity is traditionally based on Hamming distance, i.e., on how many loci are of the same value, with all those within a predefined neighborhood being included [e.g., Goldberg & Richardson, 1987]. Using this scheme, Smith and Valenzuela-Rendon [1989] found complete rule sets were maintained in two of the three cases and the failure in the third was identified as being due to the disruptive actions of crossover. That is, rules which were individually useful always produced rules which were not useful through their recombination. Booker [1982] also suggested that mating restrictions could be used such that sufficiently dissimilar rules do not recombine. Using a simple mating restriction scheme Smith and Valenzuela-Rendon [1989] found the previously unsolved problem benefited but that another no longer maintained a full solution. They concluded by suggesting that the combination of both schemes may be beneficial.

The Schema Theorem has been somewhat criticized for reasons such as the difficulty in using it to explain the dynamical or limit behaviour of GAs. Goldberg and Segrest [1987] presented a Markov chain for a simple finite population generational GA and the use of such models has remained widespread [e.g., Vose, 1999] as they enable more predictive analysis. Holland [1986b] was the first to consider using Markov chains to model LCS, the BBA in particular (reprinted in this volume). Horn et al. [1994] presented a version of Goldberg and Segrest's model to examine fitness sharing in LCS and the effects of varying the amount of interaction between two rule classes. Their model enabled them to vary the fitness ratio between the two rule types and the degree of overlap in their generalizations of the input space. By calculating the expected time to absorption of the Markov chain, they were able to show that rule maintenance times are very large even for relatively small population/rule base sizes but that this niching pressures reduces as the degree of overlap increases. Horn et al. also calculated the steady state distributions during the maintenance of both rule classes through a well-known manipulation of the absorbing Markov chain to create an ergodic chain. The degree of overlap in the generalization space was again shown to be important, causing a decrease in the probability of achieving the coverage/constitution expected from the given fitness ratio.

As noted above, these two studies used models of generational GAs as their basis. However, as described in Section 4, LCS use a steady-state GA whereby only a small percentage of the rule base is replaced per GA invocation which means that the selection pressure can be very different for example [e.g., Chakraborty et al., 1997]. Bull [e.g., 2002] presented a steady-state GA version of Goldberg and Segrest's [1987] Markov model to examine aspects of accuracy-based fitness as presented by Wilson in XCS. In comparison to a traditional strength-based fitness scheme (that is, of Holland-style systems) without fitness sharing, it was shown that XCS-type accuracy-based fitness maintained selective pressure against an incorrect rule regardless of the degree of its incorrectness, whereas the strength-based fitness scheme selected for the incorrect rule in certain cases. That is, without fitness sharing, it was shown that under strength-based fitness, a rule whose average payoff is higher than that of a correct rule can lead to the extinction of the correct rule. This phenomenon has been termed "overgeneralization" [e.g., Wilson, 1995]. Using a simple set of difference equations, Bull and Hurst [2002] showed how fitness sharing has the potential to avoid overgeneralization in both single and multi-step scenarios. Bull [2002] also included mutation into his model and showed how the accuracy-based fitness scheme appears more sensitive to the mutation rate than the strength-based scheme, a result which was previously suggested in his work on self-adaptation [Bull et al., 2000]. A simple two-step problem was also examined with the Markov chain which indicated that, under certain relationships between the rewards given for each route to the goal state, selection pressure can disappear depending upon the constituency of the rule-base. That is, using roulette-wheel selection, the effective selection pressure can vary significantly over time due to the coevolutionary nature of LCS.

5.2 Credit Assignment

The first implemented classifier system, CS-1, [Holland & Reitman, 1978] used an epochal credit assignment scheme partly inspired by Samuel's work on checkers [Samuel, 1959]. This scheme found little subsequent use (see e.g., [Smith et al., 2004] for a recent example) as it was supplanted by the Bucket-Brigade Algorithm (BBA) introduced in section 2. However, many difficulties with the BBA were soon found and alternatives suggested, e.g., [Wilson & Goldberg, 1989; Riolo, 1989; Liepins et al., 1991; Huang, 1989]. The most common form is the implicit bucket brigade described above for ZCS and XCS, wherein matching rules do not bid for control of the system, and, instead, credit is apportioned between all rules proposing a given action. After [Holland, 1986b], Westerdale [e.g., 1991; 1999] has developed a general Markov chain model for a learning entity approximating the payoff (value) of states within a given transition matrix/environment via the BBA. The aforementioned closer connection between the BBAs of ZCS and XCS and the temporal difference algorithms of the reinforcement learning literature have put credit assignment in recent LCS on firmer ground than their predecessors. For example, as noted in Section 4, XCS evolves complete maps of the entire state/action space to an estimate of value, unlike earlier systems which aim only to form a best action map, mapping each state to an action and estimate of value. The difference is significant as the more complete map potentially allows better exploration control and proper propagation of credit through the state space in the manner of reinforcement learners [Kovacs, 2004]. Indeed, convergence proofs for reinforcement learning methods require infinite updates to the estimated value of *all* state/action pairs.

Some early work also considered the use of tools emerging from the field of complexity/non-linear systems to examine LCS. Forrest and Miller [1991] cast the internal processes of Holland's LCS, in particular with a message list, as a Random Boolean network [Kauffman, 1984]. Here each node of the network is a rule and connections are formed between nodes/rules if the antecedent of one satisfies the condition of the other. By varying the specificity of rules, they show a phase transition-like dynamic exists for the emergence of self-sustaining/long inductive chains; too much or too little generalization and the LCS is unable to sustain "appropriate" internal activity. Compiani et al. [e.g., 1991] considered the fact that rule discovery and credit assignment operate over different timescales. As such, they present models of the dynamics of rule updating, for a message list of a given size, as rule discovery occurs. They find "random regimes" exist which temporarily disrupt system performance, to a significant degree, if a careful balance is not maintained against exploring newly introduced rules and exploiting existing ones.

Yates and Fairley [1994] used aspects of Evolutionary Game Theory [Maynard Smith, 1986] to show that LCS under the BBA are "evolutionary stable." That is, the rule-base of the LCS will be optimally configured for the learning task. After identifying commonalities between the features required for an evolutionary stable learning rule, i.e., one capable of finding an evolutionary stable state (ESS), and the BBA, they show a simplified LCS without a GA solving a well-known two-player game to its ESS. However, akin to the findings of Compiani et al. [1991], they note the GA is likely to disrupt the ESS, even if only temporarily.

5.3 Other Early Considerations

As noted in Section 2, LCS typically use a ternary alphabet {0,1,#} to represent rule conditions. Rule conditions are minterms, and sets of rule conditions are in Disjunctive Normal Form. This simple syntax, very similar to the binary strings used with genetic algorithms, was chosen by Holland as it was thought to be most suitable for genetic search. In particular, it was argued that the lower the cardinality of the alphabet, the higher the number of schemata and the higher the degree of implicit parallelism [Booker et al., 1989; Goldberg, 1989].

A consequence of the limited expressive power of individual rules is that sets of rules are required to represent solutions for non-trivial tasks, which introduces issues concerning the interaction of rules (i.e., competition and cooperation). Under some fitness schemes the system becomes co-evolutionary (as the fitness of one rule depends on what others exist), which complicates credit assignment and hence adaptation.

Although sets of rules using the ternary alphabet are capable of representing complex information (indeed, Holland's LCS is computationally complete [Forrest, 1985]), concerns have been raised regarding the utility of this language [e.g., Belew & Forrest, 1988; Carbonell, 1989; Grefenstette, 1989; Schuurmans & Schaeffer, 1989]. In response, Booker [1991] demonstrated a number of more expressive languages using the ternary alphabet, claiming it was the syntax of the language, rather than the cardinality of the alphabet, which was often at fault.

Whilst many continued to advocate the use of low-cardinality alphabets, the application of evolutionary methods to more complex data structures such as trees and graphs, with Genetic Programming being perhaps the best-known approach, has become widespread. Wilson suggested the use of LISP S-expressions in a classifier system [Wilson, 1994], and Lanzi subsequently studied the use of messy encodings [Lanzi, 1999a] and then S-expressions [Lanzi, 1999b] (see also [Ahluwalia & Bull, 1999]). Other representations include fuzzy logic [e.g., Valenzuela-Rendon, 1991] and neural network rules [e.g., Bull & O'Hara, 2002]. Our view is that LCS are rule-based systems, and that the vast array of possible rule languages gives them considerable expressive power, comparable to other learning methods. As always, the representation (and inductive methods) used must suit the task at hand.

Default Hierarchies (DHs) are sets of rules in which exception rules override the action of default rules (see [Holland et al, 1986] for discussions). A typical example consists of an overgeneral default rule and a set of more specific exception rules. It seems plausible that a default rule, which provides better than random performance, might be found first and that the exception rules would then be found and gradually refine the performance of the rule population as a whole. In addition to allowing such gradual refinement of representation, DHs have been seen as a means of increasing the number of solutions to a problem without increasing the size of the search space. A final advantage of DHs is that they allow more compact representations of the solution [e.g., Valenzuela-Rendon, 1989a, 1989b]. Despite these potential advantages, it has proved difficult to form and retain DHs in practice due to the complex co-evolutionary dynamics they introduce, and interest in them waned in the early 1990s. The most advanced work on the subject remains that by Smith and Goldberg [1991].

6. Foundations of Learning Classifier Systems: An Overview

The rest of this book, in keeping with the distinct areas of formal enquiry which have emerged from the field, is divided into three main sections: rule discovery, credit assignment, and problem characterization.

6.1 Rule Discovery

Jon Rowe – Population Dynamics of Genetic Algorithms. As noted above, formal understanding of evolutionary optimization techniques has progressed significantly in recent years. This contribution introduces one of the more commonly used models by which this has been done, that of Michael Vose [1999], and then introduces some extensions which are relevant to LCS thereby indicating a potentially fruitful way forward.

Lashon Booker – Approximating value functions in classifier systems. The quality of a solution to a reinforcement learning problem depends on the quality of the value function approximation (assuming one is used). This chapter notes the similarity between tile coding and a classifier system with a fixed rule population and compares the quality of the value function approximation made by the two approaches. Although the standard approach, minus the genetic algorithm, performs poorly compared to tile coding, a new hyperplane coding is introduced and the best of several variations on it is found comparable to tile coding. This represents a promising new direction for function approximation with LCS.

Larry Bull – Two Simple Learning Classifier Systems. LCS are complex and as such the production of meaningful executable models is non-trivial. This chapter presents canonical forms of each basic type, i.e., strength and accuracy-based systems, with which to examine the underlying features of each through such models.

Martin Butz et al. – Computational Complexity of the XCS Classifier System. Based on experimental results with Boolean multiplexer problems, Wilson [1998] hypothesized that the difficulty of a problem for XCS (in terms of population size and amount of experience needed) grows as a low order polynomial of the problem complexity. This chapter presents an overview of the authors’ previous work to examine these, and other, aspects of XCS formally, and establishes that k-DNF functions are PAC-learnable by XCS.

Christopher Stone and Larry Bull – An Analysis of Continuous-Valued Representations for Learning Classifier Systems. For a number of applications, particularly data mining [e.g., Wilson, 2000] and adaptive control [e.g., Hurst et al., 2002], an interval encoding has been used. This contribution considers the biases inherent within such an encoding, for both panmictic and niche-based GAs.

6.2 Credit Assignment

Jeremy Wyatt – Reinforcement Learning: A Brief Overview. LCS are now clearly identified as reinforcement learners. This contribution presents the basic mathematical framework used in the formal understanding of such techniques and discusses the various forms and extensions built from it.

John Holland – A Mathematical Framework for Studying Learning Classifier Systems. Shortly after presenting the most well-known instantiation of his LCS framework, Holland published this vision of a path to a more formal understanding of LCS ([Holland, 1986b] Reprinted with kind permission from Elsevier).

Pier-Luca Lanzi – Learning Classifier Systems: A Reinforcement Learning Perspective. This contribution demonstrates the direct connection between XCS and traditional reinforcement learning. Further, it suggests that a GA is exactly the right sort of approach to build generalizations over the input-output space of such techniques (see also [Hartmann, 1994] for a similar conclusion but from the perspective of learning difficulty in LCS).

Tim Kovacs – Rule Fitness and Pathology in Learning Classifier Systems. This chapter considers the conditions in which undesireable types of rules may prosper. Specifically, the concepts of strong overgeneral and fit overgeneral rules are introduced and linked to the structure of the value function. The prospects for such rules are investigated in both strength and accuracy-based systems, and it is suggested that accuracy-based systems have an advantage in dealing with them. This work demonstrates the existence of the above rule types using very simple tasks, to which any reinforcement learner could be applied. In doing so it demonstrates one way in which complex tasks and learners can be analysed.

Atsushi Wada et al. – Learning Classifier Systems with Convergence and Generalization. LCS for reinforcement learning incorporate function approximation through the use of rules which generalize over (aggregate) states. This chapter takes steps toward integrating LCS and standard formulations of linear function approximation in reinforcement learning. The chapter also considers convergence results. Convergence proofs exist for a number of tabular reinforcement learning methods, but no such proofs for LCS appear in the literature. As a first step, this chapter introduces a variant of ZCS to which an existing convergence proof extends. Although this version of ZCS generalizes over states, it is limited to a fixed rule population.

6.3 Problem Characterization

Anthony Bagnall and Zhanna Zatuchna – On the Classification of Maze Problems. Surprisingly, despite the many papers and many maze problems which have been presented, an overarching categorization of such tasks has been presented to date. This paper highlights features of such problems and how they can be used to group previously presented mazes and design new ones.

Tim Kovacs and Manfred Kerber – What Makes a Problem Hard for XCS? This contribution identifies four dimensions of problem complexity for XCS in the domain of Boolean functions. It suggests functions which bound the complexity of the space of functions of a given string length, and discusses how to measure the complexity of a function for XCS. Finally, it proposes a scalable Boolean test suite and argues for its use. Interested readers are referred to related work in Bernado and Ho [to appear].

7. Summary

Almost thirty years after Holland presented the Learning Classifier System paradigm, the ability of LCS to solve complex real-world problems is becoming clear. In particular, the XCS system Wilson presented ten years ago has sparked renewed interest in LCS. This article has given a brief introduction to LCS and previous formal studies of their behaviour. The rest of the book brings together work by a number of individuals who are contributing to the current formal understanding of how they achieve good performance. Future work must build on these insights to produce a coherent picture of how LCS work.

Acknowledgements

Thanks to everyone involved in this edited collection: Professor Kacprzyk for agreeing to publish the book in his series and the authors without whose efforts there would be no book. Dr. Bull would also like to thank his fellow members of the Learning Classifier Systems Group at UWE for so many interesting discussions.

References

Ahluwalia, M. & Bull, L. (1999) A Genetic Programming-based Classifier System. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela & R.E. Smith (eds) *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pp11-18. Morgan Kaufmann.

Barry, A. (2000) XCS Performance and Population Structure within Multiple-Step Environments. Ph.D. Thesis, Queens University Belfast.

Belew, R.K., & Forrest, S. (1988) Learning and Programming in Classifier Systems. *Machine Learning*, 3:193-223.

Bernado, E., Llora, X., Garrell, J.M. (2002) XCS and GALE: A Comparative Study of Two Learning Classifier Systems on Data Mining. In Lanzi, Stolzmann & Wilson (eds) Advances in Learning Classifier Systems, pp. 115-132, LNAI 2321, Springer.

Bernado, E. & Ho, T. (to appear) Domain of Competence of XCS Classifier System in Complexity Measurement Space. *IEEE Transactions on Evolutionary Computation*.

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

Booker, L. (1989) Triggered Rule Discovery in Classifier Systems. In Schaffer (ed.) *Proceedings of the International Conference on Genetic Algorithms*, pp. 265-274. Morgan Kaufmann.

Booker, L. (1982) Intelligent Behavior as an Adaptation to the Task Environment. Ph.D. Thesis, the University of Michigan.

Booker, L. (1991) Representing Attribute-based Concepts in a Classifier System. In Rawlins (ed.) *Proceedings of the First Workshop on the Foundations of Genetic Algorithms*, pp. 115-127. Morgan Kaufmann.

Booker, L., Goldberg, D.E. & Holland, J.H. (1989) Classifier Systems and Genetic Algorithms. *Artificial Intelligence*, 40: 235-282.

Bull, L. (2002) On Accuracy-based Fitness. *Soft Computing* 6(3-4): 154-161.

Bull, L. (2004)(ed.) *Applications of Learning Classifier Systems*. Springer.

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

Bull, L. Hurst, J. & Tomlinson, A. (2000) Self-Adaptive Mutation in Classifier System Controllers. In J-A. Meyer, A. Berthoz, D. Floreano, H. Roitblatt & S.W. Wilson (eds) *From Animals to Animats 6 - The Sixth International Conference on the Simulation of Adaptive Behaviour*, MIT Press.

Bull, L. & O'Hara, T. (2002) Accuracy-based Neuro and Neuro-Fuzzy Classifier Systems. In W.B.Langdon, E.Cantu-Paz, K.Mathias, R. Roy, D.Davis, R. Poli, K.Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A.C. Schultz, J. F. Miller, E. Burke & N.Jonoska (eds) *GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pp905-911. Morgan Kaufmann.

Butz, M. & Wilson, S.W. (2001) An Algorithmic Description of XCS. In *Advances in Learning Classifier Systems: Proceedings of the Third International Conference – IWLCS2000*. Springer, pp. 253-272.

Carbonell, J.G. (1989) Introduction: Paradigms for Machine Learning. *Artificial Intelligence* 40:1-9.

Chakraborty, U., Deb, K. & Chakraborty, M. (1997) Analysis of Selection Algorithms: A Markov Chain Approach. *Evolutionary Computation* 4(2): 133-167.

Compiani., M., Montanari, D. & Serra, R. (1991) Learning and Bucket Brigade Dynamics in Classifier Systems. *Physica D* 42: 202-212.

Cliff, D. & Ross, S. (1995) Adding Temporary Memory to ZCS. *Adaptive Behavior* 3(2): 101-150.

Eiben, A. & Smith, J. (2003) *Introduction to Evolutionary Computing*. Springer.

Fogel, D.B. (1992) *Evolving Artificial Intelligence*. PhD Thesis, University of California.

Forrest, S. (1985) A Study of Parallelism in the Classifier System and its Application to Classification in KL-ONE Semantic Networks. Ph.D. Thesis, University of Michigan, Ann Arbor.

Forrest, S. & Miller, J. (1991) Emergent Behavior in Classifier Systems. *Physica D* 42: 213-217.

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

Goldberg, D.E. & Richardson, J. (1987) Genetic Algorithms with Sharing for Multimodal Function Optimization. In J.J. Grefenstette (ed) *Proceedings of the 2nd International Conference on Genetic Algorithms*, pp. 41-49. Lawrence Erlbaum.

Goldberg, D.E. & Segrest P. (1987) Finite Markov Chain Analysis of Genetic Algorithms. In J.J. Grefenstette (ed) *Proceedings of the 2nd International Conference on Genetic Algorithms*, pp. 1-7. Lawrence Erlbaum.

Grefenstette, J. (1989) A System for Learning Control Strategies with Genetic Algorithms. In Schaffer (ed.) *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 183-190. Morgan Kaufmann.

Hartmann, U. (1994) On the Complexity of Learning in Classifier Systems. In Y. Davidor, H-P. Schwefel & R. Manner (eds) *Parallel Problem Solving from Nature – PPSN III*. Springer, pp. 280-289.

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

Holland, J.H. (1976) Adaptation. In Rosen & 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. (1986a). Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In Michalski, Carbonell, & Mitchell (eds) *Machine learning, an artificial intelligence approach*. Morgan Kaufmann.

Holland, J.H. (1986b) A Mathematical Framework for Studying Learning in Classifier Systems. *Physica D* 2(1-3): 307-317.

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

Holland, J.H., Holyoak, K.J., Nisbett, R.E. & Thagard, P.R. (1986) *Induction: Processes of Inference, Learning and Discovery*. MIT Press.

Horn, J., Goldberg, D.E. & Deb, K. (1994) Implicit Niching in a Learning Classifier System: Nature's Way. *Evolutionary Computation* 2(1) 37-66.

Huang, D. (1989) The Context-Array Bucket-Brigade Algorithm: An Enhanced Approach to Credit-Apportionment in Classifier Systems. In Schaffer (ed.) *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 311-316. Morgan Kaufmann.

Hurst, J., Bull, L. & Melhuish, C. (2002) TCS Learning Classifier System Controller on a Real Robot. In J.J. Merelo, P. Adamidis, H-G. Beyer, J-L. Fernandez-Villacanas & H-P. Schwefel (eds) *Parallel Problem Solving from Nature - PPSN VII,* pp588-600. Springer Verlag.

Kauffman, S. (1984) Emergent Properties of Randomly Complex Automata. *Physica D* 10: 145-156.

Kovacs, T. (1997) XCS Classifier System Reliably Evolves Accurate, Complete and Minimal Representations for Boolean Functions. In Roy, Chawdhry, & Pant (eds) *Soft Computing in Engineering Design and Manufacturing*, pp. 59-68. Springer-Verlag.

Kovacs, T. (2004) *Strength or Accuracy: Credit Assignment in Learning Classifier Systems*. Springer.

Koza, J. (1992) *Genetic Programming*. MIT Press.

Lanzi, P.L. (1999a) Extending the Representation of Classifier Conditions Part I: From Binary to Messy Coding. In Banzhaf et al. (eds.) *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 337-344. Morgan Kaufmann.

Lanzi, P.L. (1999b) Extending the Representation of Classifier Conditions Part II: From Messy Coding to S-Expressions. In Banzhaf et al. (eds.) *Proceedings of the Genetic and Evolutionary Computation Conference,* pp. 345-352. Morgan Kaufmann.

Lanzi, P.L. & Riolo, R.L. (2000) A Roadmap to the Last Decade of Learning Classifier System Research. In Lanzi, Stolzmann & Wilson (eds.) *Learning Classifier Systems. From Foundations to Applications*, pp. 33-62, LNAI 1813. Springer.

Liepins, G.E., Hillard, M.R., Palmer, R. & Rangarajan, G. (1991) Credit Assignment and Discovery in Classifier Systems. *International Journal of Intelligent Systems* 6:55-69.

Rechenberg, I. (1973) *Evolutionstrategie: Optimierung Techniser Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag.

Riolo, R.L. (1989) The Emergence of Coupled Sequences of Classifiers. In Schaffer (ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 256-264. Morgan Kaufmann.

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

Schuurmans, D. & Schaeffer, J. (1989) Representational Difficulties with Classifier Systems. In Schaffer (ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 328-333. Morgan Kaufmann.

Smith, R.E. & Valenzuela-Rendon, M. (1989) A Study of Rule Set Development in a Learning Classifier System. In Schaffer (ed.) *Proceedings of the International Conference on Genetic Algorithms*, pp. 265-274. Morgan Kaufmann.

Smith, R.E. & Goldberg, D.E. (1991) Variable Default Hierarchy Separation in a Classifier System. In Rawlins (ed.) *Proceedings of the First Workshop on the Foundations of Genetic Algorithms*, pp. 148-170. Morgan Kaufmann.

Smith, R.E., El-Fallah, A., Ravichandran, B., Mehra, R.K. & Dike, B.A. (2004) The Fighter Aircraft LCS: A Real-World, Machine Innovation Application. In L. Bull (ed) *Applications of Learning Classifier Systems*, pp113-142. Springer.

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

Valenzuela-Rendon, M. (1989a) Two Analysis Tools to Describe the Operation of Classifier Systems. PhD Thesis, University of Alabama.

Valenzuela-Rendon, M. (1989b) Boolean Analysis of Classifier Sets. In Schaffer (ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 346-353. Morgan Kaufmann.

Vose, M. (1999) *The Simple Genetic Algorithm* MIT Press.

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

Westerdale, T. (1991) Quasimorphisms or Queasymorphisms? Modeling Finite Automaton Environments. In Rawlins (ed.) *Proceedings of the First Workshop on the Foundations of Genetic Algorithms*, pp. 128-147. Morgan Kaufmann.

Westerdale, T. (1999) An Approach to Credit Assignment in Classifier Systems. *Complexity* 4:49-52.

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. (1998) Generalization in the XCS Classifier System. In Koza et al. (eds.) *Genetic Programming 1998: Proceedings of the Third Annual Conference,* pp. 322-334. Morgan Kaufmann.

Wilson, S.W. (2000) Get real! XCS with Continuous-valued Inputs. In Lanzi, P. L., Stolzmann, W., and Wilson, S. W., (eds.) *Learning Classifier Systems. From Foundations to Applications*. Springer-Verlag.

Wilson, S.W. & Goldberg, D.E. (1989) A critical review of classifier systems. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 244-255, Morgan Kauffman.

Yates, D. & Fairley, A. (1994) Evolutionary Stability in Simple Classifier Systems. In T. Fogarty (ed) *Evolutionary Computing*, pp28-37. Springer.