On Lookahead and Latent Learning in Simple LCS

Larry Bull
School of Computer Science
University of the West of England
Bristol, U.K.
+44 (0)117 2383161
Larry.Bull@uwe.ac.uk

ABSTRACT

Learning Classifier Systems use evolutionary algorithms to facilitate rule- discovery, where rule fitness is traditionally payoff based and assigned under a sharing scheme. Most current research has shifted to the use of an accuracy-based scheme where fitness is based on a rule's ability to predict the expected payoff from its use. Learning Classifier Systems which build anticipations of the expected states following their actions are also a focus of current research. This paper presents a simple but effective learning classifier system of this last type, using payoff-based fitness, with the aim of enabling the exploration of their basic principles, i.e., in isolation from the many other mechanisms they usually contain. The system is described and modeled, before being implemented. Comparisons to an equivalent accuracy-based system show similar performance. The use of self-adaptive mutation in such systems in general is then considered.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search – *backtracking, control theory, dynamic programming, graph and tree search strategies, heuristic methods, plan execution formation and execution, scheduling.*

General Terms

Algorithms, Experimentation.

Keywords

Anticipation, Genetic Algorithm, Reinforcement Learning, Self-Adaptation.

1. INTRODUCTION

Holland's Learning Classifier System (LCS) [1986] represents a form of machine learning which exploits evolutionary computing to produce inductive structures within an artificial entity. Typically, such systems use stimulus-response rules to form chains of reasoning. However, Holland's architecture has been extended to include mechanisms by which higher levels of cognitive capabilities, along the lines of those envisaged in [Holland et al., 1986], can emerge; the use of predictive modeling within LCS has been considered through alteration to the rule structure [e.g., Riolo, 1991]. Using maze tasks loosely based on those of early animal behavior experiments, it has been found that LCS can learn effectively when reward is dependent upon the ability to accurately predict the next environment state/sensory input. LCS with such 'lookahead' typically work under latent learning, i.e., they build a full predictive map of the environment without external reinforcement. LCS of this general type have gained renewed interest after Stolzmann presented the heuristics-based ACS [Stolzmann, 1998]. ACS was found to produce over-specific solutions through the workings of its heuristics and was later extended to include a Genetic Algorithm (GA)[Holland, 1975] - ACS2 [Butz & Stolzmann, 2002]. Bull [2002] presented an extension to Wilson's simple payoff-based LCS - ZCS [Wilson, 1994] - which is also able to form anticipations under latent learning. Significantly, this was the first anticipatory system to build such models through the GA alone; Riolo [1991] did not include a GA. Most current work in LCS has shifted to using accuracy as rule fitness, after Wilson presented XCS [Wilson, 1995]. Bull [2004] presented a simple accuracy-based LCS which can create such anticipations using only the GA – YCSL. In this paper, a simple payoff-based LCS which can create anticipations using only the GA is presented and explored, based on the ZCS-derived system MCS [Bull, 2005].

2. MCSL

In this paper, as in ACS and its related systems such as YACS [Gerard & Sigaud, 2002], and in [Bull, 2002; 2004], an explicit representation of the expected next environmental state is used to create a simple payoff-based anticipatory LCS which uses lookahead under latent learning - MCSL. That is, rules are of the general form:

<condition> : <action> : <anticipation>

Generalizations (#'s) are allowed in the condition and anticipation strings. Where #'s occur at the same loci in both, the corresponding environmental input symbol 'passes through' such that it occurs in the anticipated description for that input. Similarly, defined loci in the condition appear when a # occurs in the corresponding locus of the anticipation. MCSL is a Learning Classifier System without internal memory, where the rulebase consists of a number (*N*) of rules with the above form. Associated with each rule is a fitness and where the initial random population have this initialized to 10.

On receipt of an input message, the rulebase is scanned, and any rule whose condition matches the message at each position is tagged as a member of the current match set [M]. An action is then chosen from those proposed by the members of the match set at random and all rules proposing the selected action form an action set [A].

Although the use of fitness sharing for externally received payoff was fundamental to Holland's bucket brigade algorithm, it was not until Wilson introduced the action set-based scheme in ZCS that simple but effective fitness sharing in LCS became possible. MCSL, like MCS, uses the fitness sharing mechanism of ZCS, i.e., within action sets. Reinforcement consists of updating the fitness *f* of each member of the current [A] using the Widrow-Hoff delta rule with learning rate β:

$$f_j \leftarrow f_j + \beta\,(\,(P / |[A]|)\; - f_j) \qquad (1)$$

MCSL employs two discovery mechanisms, a GA and a covering operator. On each time-step there is a probability *g* of GA invocation. When called, the GA uses roulette wheel selection to determine two parent rules from the population based on their fitness. Offspring are produced via mutation (probability μ, turned into a wildcard at rate $p_{\#}$) and crossover (single point with probability χ), inheriting the parents' fitness values or their average if crossover is invoked. Replacement of existing members of the rulebase is inversely proportional to fitness, i.e., $1/(f_j+1)$, using roulette wheel selection. If no rules match on a given time step, then a covering operator is used which creates a rule with the message as its condition (augmented with wildcards at the rate $p_{\#}$) and a random action and anticipation, which then replaces an existing member of the rulebase in the usual way. It is assigned the default fitness f_0.

Hence MCSL represents a simple anticipatory LCS which relies solely upon the GA to search the space of possible generalizations; other heuristics need not be considered as pre-requisites for the effective use of a payoff-based fitness scheme. Here the term effective is taken to mean able to solve problems of low complexity whilst remaining open to close modeling; the canonical GA may be defined in much the same way. The mechanisms of MCSL are now modeled, in keeping with its philosophy, in a simple way.

3. A SIMPLE MODEL OF MCSL

The evolutionary algorithm in MCSL is a steady-state GA. A simple steady-state GA without genetic operators can be expressed in the form:

$$n(k, t+1) = n(k, t) + n(k, t)\,R(k, t)\; - n(k, t)\,D(k, t) \qquad (2)$$

where *n(k, t)* refers to the number of individuals of type *k* in the population at time *t*, *R(k, t)* refers to their probability of reproductive selection and *R(k, t)* to their probability of deletion. Roulette-wheel selection is used, i.e., *R(k, t) = f(k, t)/f(K, t)*, where *f(k, t)* is the fitness of individuals of type *k* (Equation 1) and *f(K, t)* is the total population (*K*) fitness. Replacement is inversely proportional to fitness as described above.

Table 1 shows the error 'rewards' for each of the rules considered. Those rules which experience two rewards have the average shown. Figure 1 shows the maze environment from which the errors are drawn. The maze contains two locations, one providing the LCS with input '0' and the other with input'1'. In both locations an action '0' means no move and action '1' means a move to the other location.

Table 1: Rewards for the modelled maze task.

C:A:An	Reward	C:A:An	Reward	C:A:An	Rewad
0:0:0	1000	1:0:0	0	#:0:0	500
0:0:1	0	1:0:1	1000	#:0:1	500
0:0:#	1000	1:0:#	1000	#:0:#	1000
0:1:0	0	1:1:0	1000	#:1:0	500
0:1:1	1000	1:1:1	0	#:1:1	500
0:1:#	0	1:1:#	0	#:#:#	0

The rulebase is of size *N*=400 and the initial proportions of each rule in the population are equal (*N*/18), and β=0.2. It is assumed that both inputs are presented with equal frequency, that both actions are chosen with equal frequency and that the GA fires once every four cycles (i.e., *g*=0.25). The rules' parameters are updated according to Equation 1 on each cycle.

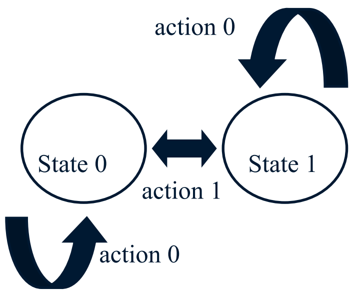
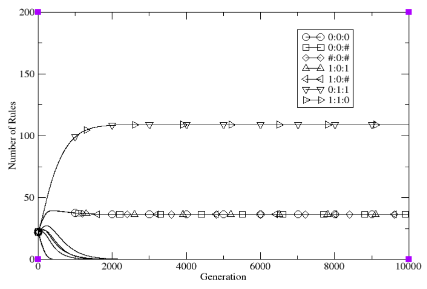


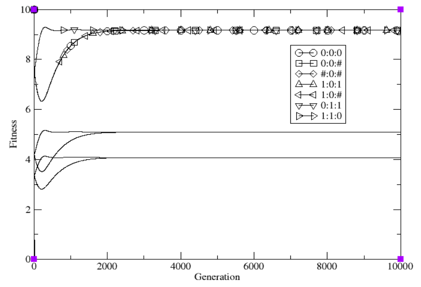
Figure 1: Simple two location maze considered.

Figure 2 shows the behaviour of the modelled MCSL on the simple maze task. Figure 2(a) shows how only the rules which accurately anticipate the next state (i.e., following their action being taken in the locations they match) exist in the final population. The rulebase is roughly divided between rules with action '0' and those with action '1' but there is no explicit pressure for a maximally general solution. Figure 2(b) shows the corresponding trajectories of the rules' fitnesses with all accurate anticipators having the same – highest - fitness. Therefore the simple payoff-based fitness scheme of MCSL results in a rulebase which completely maps the maze environment under a latent learning scenario.

This performance is roughly equivalent to, or actually slightly faster than, that produced by the simple accuracy-based system YCSL, as presented in [Bull, 2004].



(a)



(b)

Figure 2: Behaviour of model MCSL on the maze task, showing numerosity (a) and fitness (b).

4. MCSL IN T-MAZES

MCSL has been implemented and investigated using the T-maze presented in [Bull, 2002] – Woods 10. As noted above, motivation for exploring the use of learning without external reinforcement comes from early experiments in animal behaviour. Typically, rats were allowed to run around a T-maze, as shown in Figure 3, where the food cell would be empty but a different colour to the rest of the maze. The rats would then be fed in the marked location. Finally, the rats were placed at the start location and their ability to take the shortest path (go left at the T-junction in Figure 3) to the food recorded. It was found that rats could do this with around 90% efficiency. Those which were not given the prior experience without food were only 50% efficient, as expected [e.g., Seward, 1949].

Following from the model, the LCS is placed randomly in the maze and a matchset is formed. Sensory input in each location of the maze is encoded as a 16-bit binary string with two bits representing each cardinal direction. A blank cell is represented by 00, the food location (F) by 11 and trees (T) by 10 (01 has no meaning). The message is ordered with the cell directly above the LCS represented by the first bit-pair and then proceeds clockwise around it. An action is chosen at random from the matchset where there are eight possible actions (cardinal moves) and the LCS can move into any one of the surrounding eight cells on each discrete time step, unless occupied by a tree or it is the food location (this avoids creating a sensory ambiguity). All rules which propose the chosen action are updated. One further mechanism is incorporated for this harder task (after [Bull, 2002]): the first *N* random rules of the rulebase have their anticipation created using cover (with #'s included as usual) in the first [A] of which they become a member. This goes some way to make " ... good use of the large flow of (non-performance) information supplied by the environment." [Holland, 1990]. Rules created under the cover operator also receive this treatment. In this way the GA explores the generalization space of the anticipations created by the simple heuristic.

T	T	T	T	T	T	T
T	F	T	T	T		T
T		T	T	T		T
T						T
T	T	T		T	T	T
T	T	T		T	T	T
T	T	T	T	T	T	T

Figure 3: Woods 10 maze.

Initial results with *N*=5000, $p_{\#}$=0.6, β=1.0, *g*=1.0, χ=0.5 and μ=0.04 (after [Bull, 2004]) find MCSL is unable to produce a full model of Woods 10 (not shown). Indeed, the system appears to predict a low number of actions, with increasing specificity, and the rule with the highest numerosity in those few [A] rarely anticipates the next state correctly.

Under the operations of the GA within ZCS there is a reproduction cost such that parents give half of their fitness to their offspring. No explanation for this mechanism is given in [Wilson, 1994] but it has been suggested that it produces a generalization pressure within such systems. That is, once a rule has reproduced, it and its offspring are much less likely to be picked again under the global GA until their niche occurs, at which point they are assigned a new fitness appropriate for the current numerosity. The more general rule is updated faster. The first point is fundamental to the way in which fitness sharing avoids overgeneral rules since it removes any advantage in difference between niche payoff levels [Bull, 2005]; the payoff available to individual rules becomes the same in all niches once numerosities have been adjusted appropriately by the GA.

Figure 4 shows how the mechanism works well within MCSL, after [Bull, 2002]. Eight actions were maintained throughout in this case (not shown).

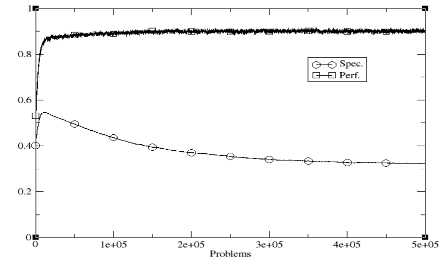
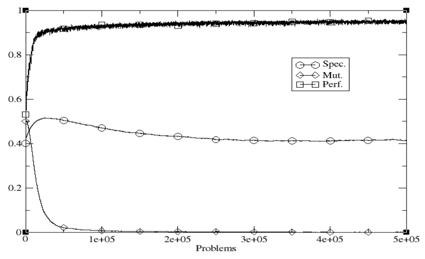


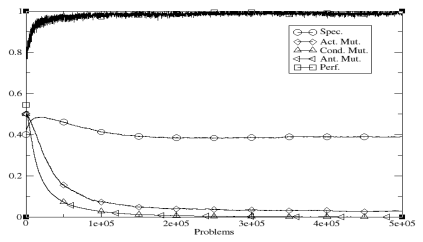
Figure 4: Behaviour of MCSL in Woods 10, equal to that of an equivalent accuracy-based system in [Bull, 2004].

5. SELF-ADAPTIVE MUTATION

In previous work [e.g., Bull et al., 2000], a single self-adapting mutation rate parameter was added to each rule in LCS. That is, each rule has its own mutation rate μ, stored as a real number and initially seeded uniform randomly in the range [0.0,1.0]. This parameter is passed to its offspring either under recombination or directly. The offspring then applies its mutation rate to itself using a Gaussian distribution, i.e., $\mu_i' = \mu_i e^{N(0,1)}$, before mutating the rest of the rule at the resulting rate. Figure 5(a) shows how the approach can be successfully used within MCSL in Woods 10 using all others the parameters as before. There is a significant rise (T-test, $P<0.05$) in the specificity of the solution produced however compared with the fixed single mutation rate of $\mu=0.04$.



(a)



(b)

Figure 5: Behaviour of MCSL in Woods 10 with a single self-adapting mutation rate (a) and three (b).

The creation of an anticipatory system through the GA alone means the evolutionary process is designing rule structures of increased complexity in comparison to the traditional stimulus-response rules. It may therefore be beneficial to increase the freedom of the mutation operator to search the sub-spaces of the different parts of the rule encoding at different rates; improvements in performance may be possible with separate self-adapting mutation rates for the condition, action and anticipation. This has been explored with each mutation rate adapting as before, as shown in Figure 5(b). The rate for the action component adapts differently to those of the condition and anticipation. Specificity is still higher than with the fixed rate as a consequence however, but less than that seen with the single adapting parameter. A more converged and accurate solution is produced than in the other two systems. No significant increase in learning speed is seen, again solutions appear to settle after around 200,000 problems, which is faster than with the fixed rate which typically settle to a solution after around 300,000 problems; a slightly more specific solution is learnt more quickly.

Results from using both mechanisms within the equivalent simple accuracy-based system also show an increase in specificity. This is to such a degree that a larger population is required to enable the evolution of a solution wherein the fittest rule in each [A] accurately predicts the next state in all states (not shown).

A full version of this paper is available as Technical Report UWELCSG07-002 from http://www.cems.uwe.ac.uk/lcsg

6. REFERENCES

Bull, L. (2002) Lookahead and Latent Learning in ZCS. In W.B.Langdon et al. (eds) *GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, pp897-904.

Bull, L. (2004) Lookahead and Latent Learning in a Simple Accuracy-based Learning Classifier System. In X. Yao et al. (eds) *Parallel Problem Solving from Nature - PPSN VIII*. Springer Verlag, pp1042-1050.

Bull, L. (2005) Two Simple Learning Classifier Systems. In L. Bull & T. Kovacs (eds) *Foundations of Learning Classifier Systems*. Springer, pp63-90.

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

Butz, M. & Stolzmann, W. (2002) An Algorithmic Description of ACS2. In P-L. Lanzi, W. Stolzmann & S.W. Wilson (eds) *Advances in Learning Classifier Systems: IWLCS 2001*. Springer, pp211-230.

Gerard, P. & Sigaud, O. (2001) YACS: Combining Dynamic Programming with Generalization in Classifier Systems. In P-L. Lanzi, W. Stolzmann & S.W. Wilson (eds) *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop.* Springer, pp52-69.

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

Holland, J.H. (1986) Escaping Brittleness. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (eds) *Machine Learning: An Artificial Intelligence Approach, 2*. Morgan Kauffman, pp48-78.

Holland, J.H. (1990) Concerning the Emergence of Tag-Mediated Lookahead in Classifier Systems. *Physica D* 42:188-201.

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

Riolo, R. (1991) Lookahead Planning and Latent Learning in a Classifier System. In J-A. Meyer & S.W. Wilson (eds.) *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behaviour*. MIT Press, pp316-326

Seward, J.P. (1949) An Experimental Analysis of Latent Learning. *Journal of Experimental Psychology* 39: 177-186.

Stolzmann, W. (1998) Anticipatory Classifier Systems. In J.R. Koza (ed) *Genetic Programming 1998: Proceedings of the Third Annual Conference*. Morgan Kaufmann, pp658-664.

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):149-177.