ZCS Redux

Larry Bull larry.bull@uwe.ac.uk
Faculty of Computing, Engineering and Mathematical Sciences, University of the West of England, Bristol BS16 1QY, UK

Jacob Hurst jacob.hurst@uwe.ac.uk
Faculty of Computing, Engineering and Mathematical Sciences, University of the West of England, Bristol BS16 1QY, UK

Abstract

Learning classifier systems traditionally use genetic algorithms to facilitate rule discovery, where rule fitness is payoff based. Current research has shifted to the use of accuracy-based fitness. This paper re-examines the use of a particular payoff-based learning classifier system – ZCS. By using simple difference equation models of ZCS, we show that this system is capable of optimal performance subject to appropriate parameter settings. This is demonstrated for both single- and multistep tasks. Optimal performance of ZCS in well-known, multistep maze tasks is then presented to support the findings from the models.

Keywords

Genetic algorithms, learning classifier systems, fitness sharing, generalization, memory.

1 Introduction

Learning classifier systems (LCS) (Holland, 1976) traditionally use *genetic algorithms* (GAs) (Holland, 1975) to discover new rules/generalizations for a given problem space. In the first implementation of LCS, rule fitness was viewed as a predictor of expected future payoff (Holland and Reitman (1978), based on Samuel (1959)). Payoff was only given to rules that predicted a reward no greater than the actual value received from the environment "to reflect their accuracy in anticipating this reward" (Holland and Reitman, 1978). Under Holland's (1986) later formalism, rule fitness is purely payoff based and serves as a measure of a given rule's "usefulness" in receiving external payoff. Current research has again focused on the use of accuracy in rule predictions as the fitness measure, after Wilson (1995) introduced the fundamentally accuracy-based XCS. In particular, XCS aims to avoid problematic overgeneral rules that receive a high optimal payoff for some inputs but are suboptimal for other, lower payoff inputs. Since their average payoff is higher than that for the optimal rules, in the latter case, the overgenerals tend to displace them, leaving the LCS suboptimal. However, the payoffs received by overgeneral rules typically have high variance (they are inaccurate predictors) and so have low fitness in XCS.

This paper revisits Wilson's (1994) ZCS architecture (presented by Wilson immediately prior to XCS), which "keeps much of Holland's original framework but simplifies it to increase understandability and performance" (ibid.). Results from developing a set of simple difference equation models of ZCS are presented here, which show its potential to perform optimally, avoiding overgenerals through its use of fitness sharing.

Using a single-step task, it is further shown that the learning rate plays an important role and that the modeled ZCS can be robust to changes in the task. A multistep version of the model is then presented, which again shows its potential to perform optimally. Based on these findings, optimal performance is presented for the first time in tasks previously used in the literature to examine ZCS: `Woods1`, `Woods14`, and `Woods101`. ZCS's original suboptimal performance in these environments, using the parameters in Wilson (1994), forms the basis of much of its criticism and that of payoff-based fitness in general.

The paper is arranged as follows. The next section briefly describes ZCS. Section 3 presents the development of a model of ZCS in single-step tasks and the results of its use. Section 4 presents a multistep version of the model with results. In Section 5, new experimental results with ZCS are presented. Finally, all results are discussed.

2 ZCS: A Zeroth Level Learning Classifier System

ZCS is a Michigan-style LCS without internal memory, which periodically receives a binary encoded input from its environment. The system determines an appropriate response based on this input and performs the indicated action, usually altering the state of the environment. Desired behavior 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 ZCS rule base consists of a population of P condition-action rules or "classifiers." The rule condition is a string of characters from the ternary alphabet 0,1,#, where # acts as a wildcard allowing generalization. The action is represented by a binary string, and both conditions and actions are initialized randomly. Also associated with each classifier is a fitness scalar initialized to a predetermined value S_0.

On receipt of an input message, the rule base 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 selected from those advocated by the rules comprising [M]. In ZCS, this is performed by a simple roulette-wheel selection policy based on fitness. Once an action has been selected, all rules in [M] that advocate this action are tagged as members of the action set [A], and the system executes the action.

Reinforcement in ZCS consists of redistributing payoff between subsequent action sets. A fixed fraction β of the fitness of each member of [A] at each time-step is placed in a "common 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 predetermined discount factor γ. If a reward is received from the environment, then a fixed fraction β of this value is distributed evenly among the members of [A] divided by their number. Finally, on each time-step, a tax is imposed on the members of [M] that do not belong to [A] 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 ZCS represents a change to the traditional LCS bucket brigade algorithm (Holland, 1986), since there is no concept of a rule "bid," specificity is not considered explicitly, and bucket payments are reduced by $1 - \gamma$ on each step. As will be shown here, ZCS uses fitness sharing within the external payoff niche/matchset which Holland's algorithm included in an early description (Holland, 1985) but not in the more well-known presentation (Holland, 1986) (being mentioned as a possibility in Holland et al. (1986)). A loose similarity between the bucket brigade and Q-learning

(Watkins, 1989) is also noted by Wilson (see Dorigo and Bersini (1994) for a general comparison). However, as noted in Sutton and Barto (1998), the bucket brigade is more akin to Sutton's (1996) simple temporal difference learning algorithm called Sarsa (see Singh et al. (2000) for a recent analysis of Sarsa).

ZCS employs two discovery mechanisms: a GA operating over the whole rule set at each instance (panmictic) 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.

If on some time-step, [M] is empty or has a combined fitness of less than f times the population average, then a covering operator is invoked. A new rule is created with a condition that matches the environmental message and a randomly selected action. The rule's condition is then made less specific by the random inclusion of #'s at a probability of 0.33 per bit. The new rule is given a fitness equal to the population average and inserted into the population, overwriting a rule selected for deletion as before.

Typical parameters used are: Rule base P = 400, initial rule fitness S_0 = 20.0, learning rate β = 0.2, discount factor γ = 0.71, tax τ = 0.1, cover trigger ϕ = 0.5, GA rate per time step ρ = 0.25, crossover rate χ = 0.5, and per bit mutation rate μ = 0.002.

Wilson (1994) showed ZCS could almost solve the maze tasks `Woods1` and `Woods7`, the latter being non-Markov and hence unsolvable without additional memory mechanisms. Cliff and Ross (1995) showed its inability to solve longer multistep tasks, proposing forms of overgeneralization as being the cause, as well as implementing the memory register suggested in Wilson (1994). Bull (1998, 1999) used it in a multiagent context to represent the traders of a simplified market. Tomlinson and Bull (1998, 1999a, 1999b) incorporated rule linkage for non-Markov domains. Cao et al. (1999, 2000) used ZCS to control road traffic junctions. Ahluwalia and Bull (1999) used numerical S-expressions for actions, allowing the action/output to be a function of the input value. Bagnall (2000) modeled agents in a simulated electricity market with ZCS. Bull (2001a) incorporated lookahead and latent learning mechanisms. The self-adaptation of mutation (Bull and Hurst, 2000; Bull et al., 2000) and other parameters (Hurst and Bull, 2001) have also been added.

The mechanisms of ZCS are now modeled, in keeping with the philosophy of ZCS itself, in a simple way to gain a better understanding of its working.

3 A Simple Model of ZCS

There has been some prior work modeling the role of the GA within LCS explicitly. Smith and Valenzuela-Rendon (1989) presented a difference equation model of an infinite population, generational GA for an LCS in a single-step task, finding that niching or fitness sharing was necessary to facilitate task coverage in difficult cases. Horn et al. (1994) presented a simple Markov chain model of a generational GA for an LCS working in a single-step task, also examining the effects of fitness sharing and presenting analysis of niche maintenance times. A steady-state GA version of their model was presented by Bull (2001b) to highlight the benefits of accuracy-based over payoff-based schemes, also noting their different sensitivities to mutation rates and behavior in multistep tasks.

Table 1: Reward payoffs for the single-step task considered.

Input	Action	Payoff
1	1	1000
1	0	8
0	1	1000
0	0	3000
#	0	1504
#	1	1000

Bull (2001b) did not consider fitness sharing in his models. As noted above, the role of fitness sharing was examined in the other models of payoff-based LCS and found to have a significant effect. It is included in ZCS in a particularly elegant way, based on Wilson's (1987) Boole system. Wilson (1987) presented simple equations for the effects of fitness sharing in Boole, which suggested that at equilibrium, all rules will tend have the same fitness value and that the number of rules per action set is proportional to the received payoff for that action. A simple model of ZCS is now developed that highlights the role of fitness sharing in this payoff-based LCS, as was done in some of the prior work noted above. The bucket brigade algorithm is then included, which leads to new results.

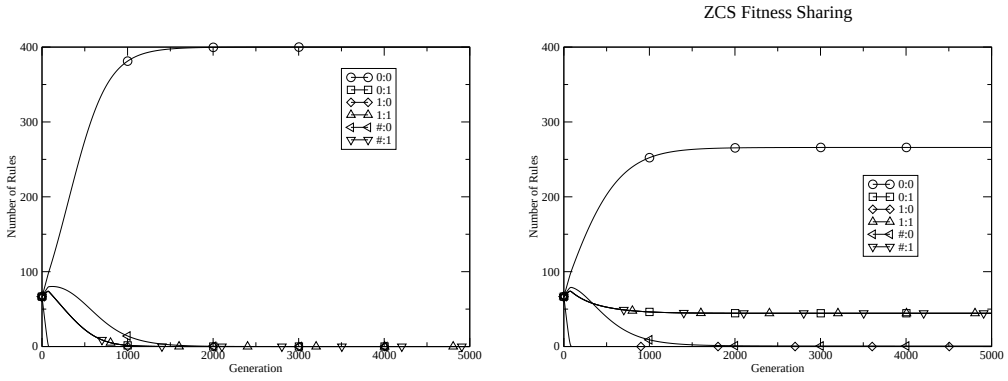
3.1 A Steady-State GA with ZCS Fitness Sharing

A simple, steady-state GA without genetic operators can be expressed in the form:

$$n(k,t+1) = n(k,t) + n(k,t)R(k,t) - n(k,t)D(k,t) \tag{1}$$

where $n(k,t)$ refers to the number of individuals of type k (typically, binary strings) in the population at time t, $R(k,t)$ refers to their probability of reproductive selection, and $D(k,t)$ to their probability of deletion. Roulette-wheel selection is used in ZCS, i.e., $R(k,t) = f(k,t)/f(P,t)$, where $f(k,t)$ is the fitness of individuals of type k and $f(P,t)$ is the total population fitness. Replacement is inversely proportional to fitness as noted above.

Table 1 shows the payoffs for the single-step task with a single-bit condition and single-bit action considered here. It is assumed that both inputs are presented with equal frequency. The last two entries in Table 1 show the expected payoff for the general rules of the task, following Kovacs (2000), i.e., the fitness of a general rule is the average of the payoffs it receives (also used in Bull (2001b)). It can be seen that under this scheme, for input 1, the general rule #:0 has a higher payoff than the correct rule 1:1 and so is most likely to be selected; #:0 is an overgeneral rule that would cause suboptimal performance. The progress of all six rules is examined here. In the simplest case, $f(k,t) = payoff(k,t)$, hence $f(0:0,t) = 1000$, etc., and $f(\#:0,t) = (f(0:0,t) + f(1:0,t))/2$, etc. That is, fitness is not updated as a running average as is usually the case in the bucket brigade. Therefore, with equations of the general form shown in Equation (1), the expected proportions of each rule type in the next generation can be determined; by specifying the initial proportions of each rule in the population $(P/6)$, it is possible to generate the trajectory of their proportions over succeeding generations. Note, partial individuals are allowed, and hence it is, in effect, an infinite population model.



(a) (b)

Figure 1: Behavior of model LCS on the task in Table 1 without and with fitness sharing.

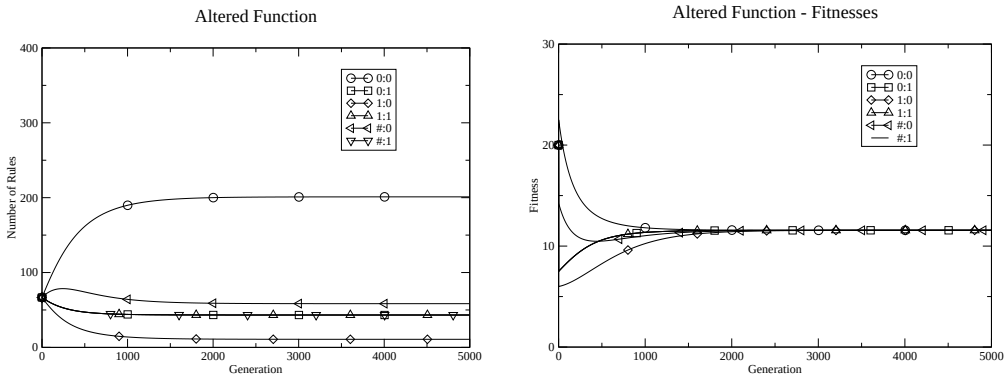
Figure 1(a) shows the resulting behavior of the steady-state GA/LCS using the same parameters as those in Wilson (1994), i.e., $P = 400$ and $S_0 = 20.0$. Figure 1(b) shows the behavior of the same system including the fitness sharing mechanism of ZCS such that payoff is divided by the size of the corresponding [A]:

$$f(a,t) = payoff(a,t)/n(a,t) \tag{2}$$

where $payoff(a,t)$ is the payoff received for taking action a, $f(a,t)$ is the fitness of rules advocating action a, and $n(a,t)$ is the number of matching rules that advocate the same action at time step t. Hence, for example, $f(0:0,t) = 3000/(n(0:0,t) + n(\#:0,t))$, etc. The general rules are handled as before.

It can be seen that in the first case, the rule base converges to a single type of high payoff rule – 0:0. However, with the inclusion of fitness sharing, the rule base contains four types of rule: 0:0, 0:1, 1:1, and the general rule #:1. In all cases, the rules have the same fitness value (not shown), expected payoff being indicated by the relative number of rules of a given type. The proportions of rules in Figure 1(b) means that, under the roulette-wheel action selection scheme used in ZCS, the system will (on average) give optimal performance, since for input 0, there will be more rules of type 0:0 in [M] than any other and an equal number of rules 1:1 and #:1 for input 1. It is also interesting to note that the simple ZCS-like system effectively contains a map of the problem, including its generalization $payoff(0:1,t) = payoff(1:1,t)$.

This result is contrary to many of the discussions on the problems of payoff-based fitness, namely their inability to maintain rules for different payoff levels (e.g., Wilson (1994)) and their tendency to promote overgenerals (Cliff and Ross, 1995; Wilson, 1995; Kovacs, 2000). So, why has this simple model of ZCS not suffered from these problems? As stated above, the basic premise for the persistence of overgenerals within an LCS is that such rules receive a higher payoff in a niche for which they are correct than the payoff possible in another niche for which they are incorrect, where their average payoff is higher than that of the second niche. But under fitness sharing, all rule fitnesses tend to the same value (as discussed in Wilson (1987)). Hence, in the model, the payoff received by a potentially overgeneral rule in one niche is eventually insignificantly



(a) (b)

Figure 2: Behavior of model LCS on the slightly altered task from that in Table 1 (1:0 = 800).

different from that which it receives in any other, the degree of occupancy of any given niche being altered by the actions of the GA. Therefore the seemingly overgeneral rule #:0 did not have a significantly higher fitness than 1:0 or #:1, because it did not receive a payoff of 3000 in the input 0 niche. Similarly, appropriate rules for each input, despite the different payoff levels, are maintained by the same mechanism.

The situation is not as simple as this analysis implies. By increasing the small payoff entry in Table 1 to a value closer to the other for input 1 (e.g., 800), the overgeneral rule #:0 is found to obtain a greater proportion of the ZCS rule base than any correct rules under input 1 (Figure 2). This model, as with all previous LCS GA models mentioned above, assumes rule fitnesses converge instantaneously (akin to Pittsburgh-style LCS (Smith, 1980)). However, ZCS uses a form of temporal difference learning to update rule fitnesses at a speed/weight determined by the learning rate. It is well known from temporal difference learning that suboptimal solutions can arise if the learning rate and/or discount rate are incorrect for the given task (e.g., Sutton and Barto (1998)). A more detailed model of ZCS is needed to examine this aspect of its behavior.

3.2 The (Implicit) Bucket Brigade

Under the bucket brigade of ZCS, rules chosen as the action pay β of their fitness into the bucket or pay a tax if they are not, i.e.,

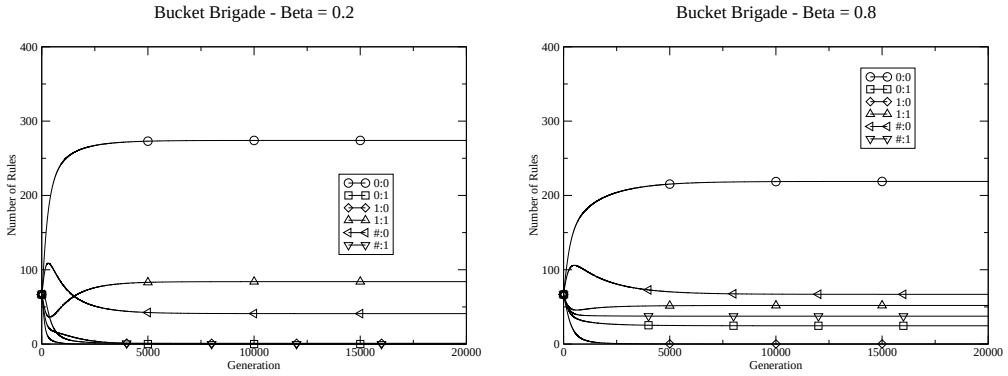
$$f(a,t+1) = S(a,t)(f(a,t) - \beta f(a,t)) + (1 - S(a,t))(f(a,t) - \tau f(a,t)) \tag{3}$$

where $S(a,t)$ refers to the probability of selecting action a:

$$S(a,t) = \frac{\sum_{i=0}^{\alpha} f(i,t)n(i,t)}{\sum_{j=0}^{\varphi} f(j,t)n(j,t)} \tag{4}$$

where there are α classes of rule with action a and j different actions in a given [M]. Any external reward is then shared equally among the action set, decreased by β:

$$f(a,t+1) \leftarrow f(a,t+1) + S(a,t)(\beta payoff(a,t)/n(a,t)) \tag{5}$$



(a) (b)

Figure 3: Behavior of model ZCS on the slightly altered task from Table 1 with different learning rates.

Therefore, for example, the full fitness difference equation for rule 0:0 is:

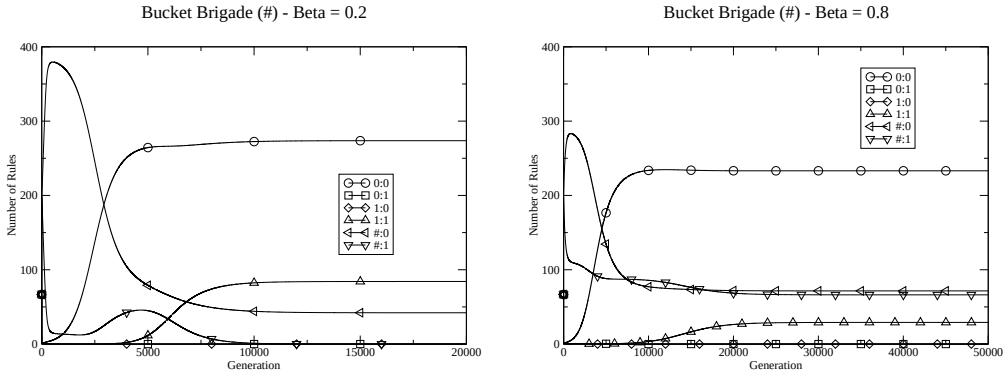
$$f(0:0,t+1) = [(n(0:0,t)f(0:0,t) + n(\#:0,t)f(\#:0,t))/F_0)((f(0:0,t) - \beta f(0:0,t))]$$
$$+[(1 - ((n(0:0,t)f(0:0,t) + n(\#:0,t)f(\#:0,t))/F_0))((f(0:0,t) - \tau f(0:0,t))]$$
$$+[(n(0:0,t)f(0:0,t) + n(\#:0,t)f(\#:0,t))/F_0)(\beta payoff(0,t)/((n(0:0,t) + n(\#:0,t))]$$

where F_0 refers to the total fitness of the matchset for input 0, i.e.,

$$F_0 = f(0:0,t)n(0:0,t) + f(0:1,t)n(0:1,t) + f(\#:0,t)n(\#:0,t) + f(\#:1,t)n(\#:1,t)$$

Figure 3(a) shows the behavior of the ZCS model using the above equations to update rule fitnesses on the task in Table 1 with the payoff for input 1 and action 0 altered to 800. All relevant parameters are as in Section 2, and two presentations of input 0 and 1 are assumed before the GA fires ($\rho = 0.25$). The rules containing generalizations are, of course, updated twice as often as the others, being members of each [M]. It can be seen that this more complete ZCS model solves the task optimally (as defined in Section 3.1), converging on the two specific rules for the task – 0:0 and 1:1. Initially, more rules of the type #:0 appear, but these decrease as the GA begins to exert its load balancing effect. Figure 3(b) shows the effect of increasing the learning rate from 0.2 to 0.8. Here the overgeneral rule #:0 maintains a greater proportion of the rule base than any correct rules for input 1. Thus with a greater consideration of the last payoff received, the modeled ZCS converges on the suboptimal solution (as it did under Equation (1)). That is, the modeled ZCS can avoid problematic overgeneral rules depending upon the rate at which rule fitnesses are updated.

In the above experiments, the rule base was initialized with an equal number of each rule type $(P/6)$. Figure 4 shows results on the same task altering this balance in favor of the rules containing generalizations: the initial rule base contained exactly one copy of each specific rule and 398 rules equally divided between #:0 and #:1. Figure 4(a) shows how the modeled ZCS still avoids overgeneralization using the learning rate identified as beneficial in Figure 3(a), although with some delay. Similarly, in Figure 4(b), the higher learning rate again results in a rule base that would give suboptimal



(a) (b)

Figure 4: Behavior of model ZCS with altered initialization and different learning rates.

Table 2: Reward payoffs for second part of the first dynamic single-step task considered.

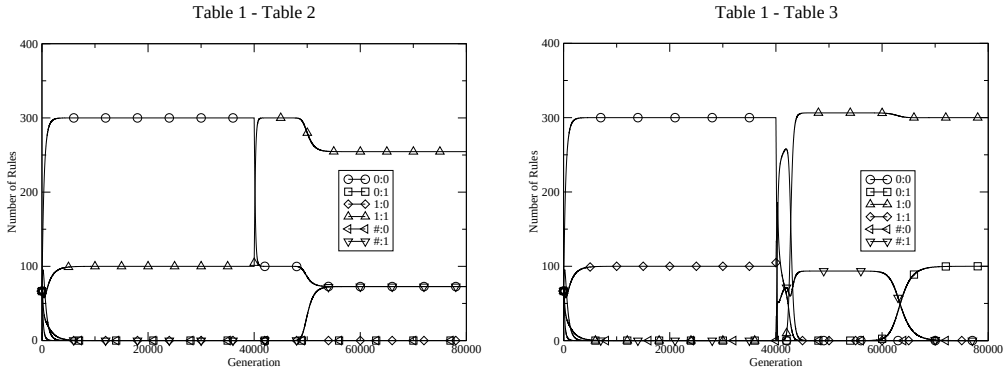
Input	Action	Payoff
1	1	3000
1	0	8
0	1	1000
0	0	1000

performance under ZCS's action selection scheme. Hence it would appear that, in the simple model presented here at least, the speed at which rules update their fitnesses is critical to the fitness sharing process. This is perhaps not unexpected given that the number of rules within each [A] is (potentially) constantly altered under the actions of the GA during the learning period.

Another aspect of fitness sharing and its consequences for ZCS are now examined.

3.3 Nonstationary Tasks

Many real-world problems, such as on-line process control, are dynamic and hence machine learning techniques must be able to operate under changing conditions if they are to be used effectively in such domains. Fitness sharing has been used within traditional evolutionary optimization for such nonstationary problems (e.g., Anderson (1991)) since the maintained diversity gives the potential for individuals/solutions to already exist near the newly located optimum. ZCS's potential robustness to change through its use of fitness sharing has also been examined using the model from Section 3.2. Figure 5(a) shows the behavior of the modeled ZCS as the task was switched from that in Table 1 to that in Table 2 after 40,000 GA generations, with all parameters as in Section 2. It can be seen that after switching the payoffs for 0:0 and 1:1, the model responds relatively quickly, around 500 GA generations (1000 system cycles), adjusting the proportion of the two rules appropriately. The number of general rules #:1 also increases after the change to match the number of 0:0 rules, as expected from Table 2, this



(a) (b)

Figure 5: Behavior of model ZCS in the nonstationary tasks.

Table 3: Reward payoffs for the second part of the second dynamic single-step task considered.

Input	Action	Payoff
1	1	1000
1	0	3000
0	1	1000
0	0	8

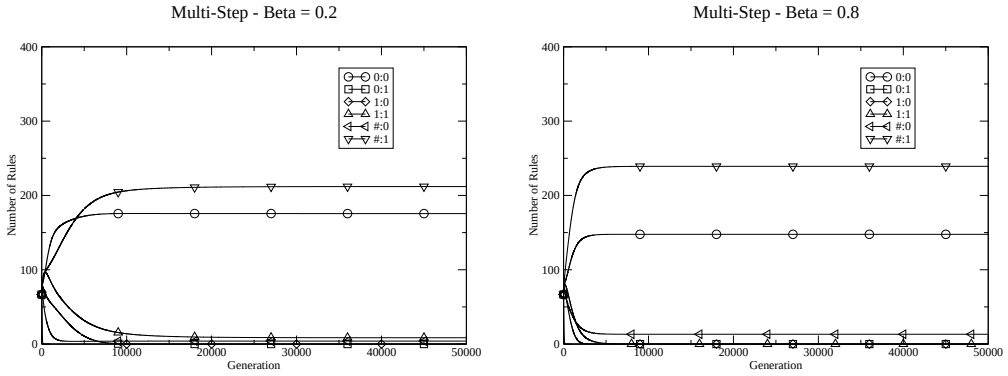
rule representing a succinct optimal solution to the new task.

Figure 5(b) shows the behavior of the same system when the task is switched from that in Table 1 to that in Table 3 after 40,000 generations. Here the payoffs for 0:0 and 1:0 are switched. It can be seen that the modeled ZCS takes much longer to respond to the change, around 5000 GA generations, since an optimal solution is represented by rules with very low presence in the rule base for the first task. It eventually converges on an appropriate number of rules 1:0 and 0:1. Therefore the modeled ZCS can be robust to nonstationary tasks, depending upon the type of change that occurs.

In summary, the modeled ZCS has been shown able to perform optimally in various simple, single-step environments with an appropriate learning rate. Its use in multistep environments can also be considered through an extension to the model.

4 A Model of ZCS in Multistep Environments

In this paper, a two-step delayed reward task is considered, with input 0 on the first step and input 1 on the second. Hence on the first step of every trial, all rules in [M] for input 0 adjust their fitnesses according to Equation (3). Then all rules in [M] for input 1 adjust their fitnesses according to Equation (3), where the fitnesses used by the general rules are those resulting from their change in the first matchset. Next the value B of the



(a) (b)

Figure 6: Behavior of model ZCS in a multistep task under different learning rates.

Table 4: Reward payoffs for the two-step task considered.

Action1	Action2	Payoff
1	1	1000
1	0	8
0	1	1000
0	0	8

bucket from the second step is calculated, i.e.,

$$B = \sum_{i=0}^{\alpha} S(i,t)\beta f(i,t)n(i,t) \tag{6}$$

Then the appropriate external payoff is calculated and paid to the rules of the second matchset:

$$f(a,t+1) \leftarrow f(a,t+1) + S(a,t)\beta payoff(a',t)/n(a,t) \tag{7}$$

where $payoff(a',t)$ in the two-step case considered here are:

$$\begin{aligned} payoff(0,t) &= S(0,t-1)payoff(0-0,t) + S(1,t-1)payoff(1-0,t) \\ payoff(1,t) &= S(0,t-1)payoff(0-1,t) + S(1,t-1)payoff(1-1,t) \end{aligned} \tag{8}$$

where $S(0,t-1)$ is the probability an action 0 was taken on the first step (Equation (4)), and $payoff(0-1,t)$ is the payoff received for the system taking action 0 on the first step and action 1 on the second step, etc.

The value in the bucket is then discounted and shared equally among the rules on the first step:

$$f(a,t+1) \leftarrow f(a,t+1) + S(a,t)\gamma\beta/n(a,t) \tag{9}$$

Again, care must be taken in the case of the general rules, and two passes through the production system are assumed before the GA fires. All parameters are as in Section 2.

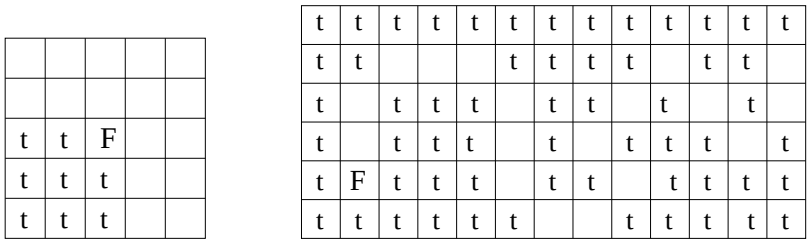


Figure 7: Two multistep mazes used to test ZCS, (a) `Woods1` and (b) `Woods14`.

Table 4 shows the payoff scheme used in Figure 6. Figure 6(a) shows that the modeled ZCS converges on an optimal solution represented by the general rule #:1 with a slightly smaller number of the specific rule 0:0, which can be used as part of an optimal solution (Table 4). Figure 4(b) shows the effect of increasing the learning rate to 0.8. Here it can be seen that an optimal solution is again obtained but that the higher learning rate has increased the proportion of the rule base occupied by the succinct solution #:1. That is, in the simple multistep task, a higher learning rate has produced a potentially less varying solution under the (probabilistic) action selection scheme of ZCS. This is in contrast to the single-step task results in Section 3, where a higher learning rate was found to be detrimental, implying that for sufficiently different problems, different rates of rule fitness update are required for effective fitness sharing and/or system behavior to occur.

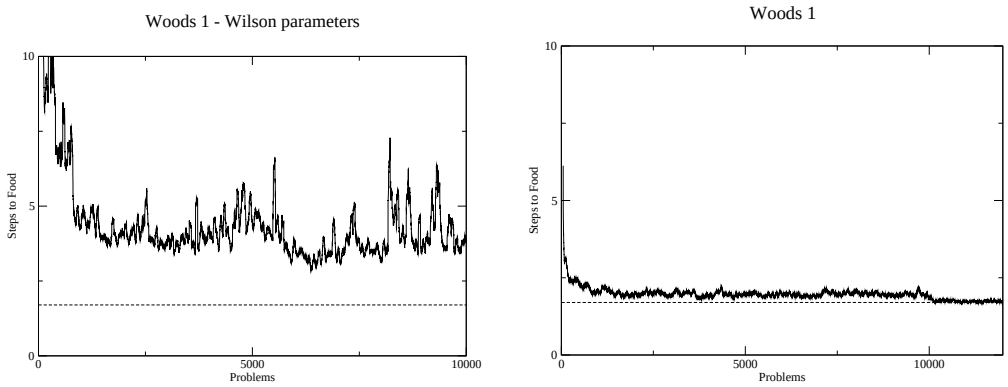
The following section considers the performance of ZCS itself in a number of mazes previously used to test its performance, in light of the findings from the simple models; it will be shown that ZCS, with appropriate parameters, can solve the tasks optimally.

5 ZCS in Multistep Environments

5.1 `Woods1`

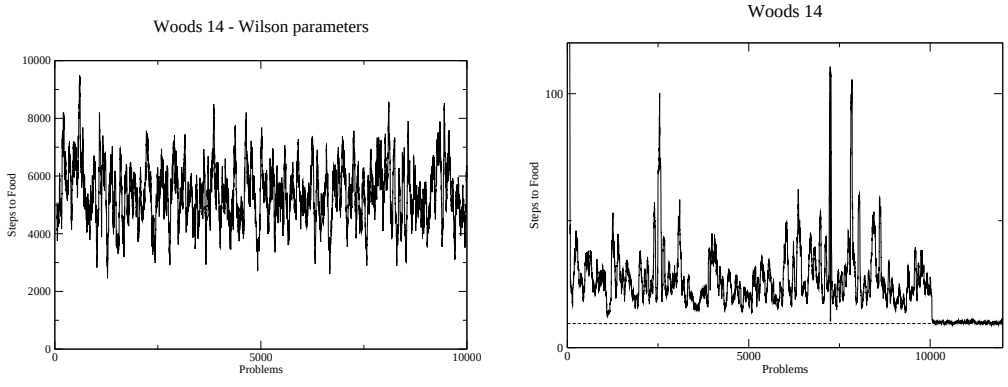
As noted in Section 2, Wilson (1994) introduced two multistep maze environments with which to examine the performance of ZCS: `Woods1` and `Woods7`. `Woods1` is a two-dimensional rectilinear 5x5 toroidal grid. Sixteen cells are blank, eight contain trees, and one contains food. ZCS is used to develop the controller of a simulated robot/animat that must traverse the map in search of food. It is positioned randomly in one of the blank cells and can move into any one of the surrounding eight cells on each discrete time step, unless occupied by a tree. If the animat moves into the food cell, the system receives a reward from the environment (1000), and the task is reset, i.e., food is replaced and the animat randomly relocated (Figure 7(a)).

On each time step, the animat receives a sensory message that describes the eight surrounding cells. The message is encoded as a 16-bit binary string with two bits representing each cardinal direction. A blank cell is represented by 00, food (F) by 11, and trees (t) by 10 (01 has no meaning). The message is ordered with the cell directly above the animat represented by the first bit pair and proceeds clockwise around the animat.



(a) (b)

Figure 8: Performance of ZCS in `Woods1` under different parameters.



(a) (b)

Figure 9: Performance of ZCS in `Woods14` under different parameters.

Here, as in Wilson (1994), the trial is repeated 10,000 times and a record is kept of a moving average (over the previous 50 trials) of how many steps it takes for the animat to move into a food cell on each trial. If it moves randomly, Wilson calculates performance at 27 steps per trial, while the optimum is 1.7 steps. All results presented are the average of ten runs.

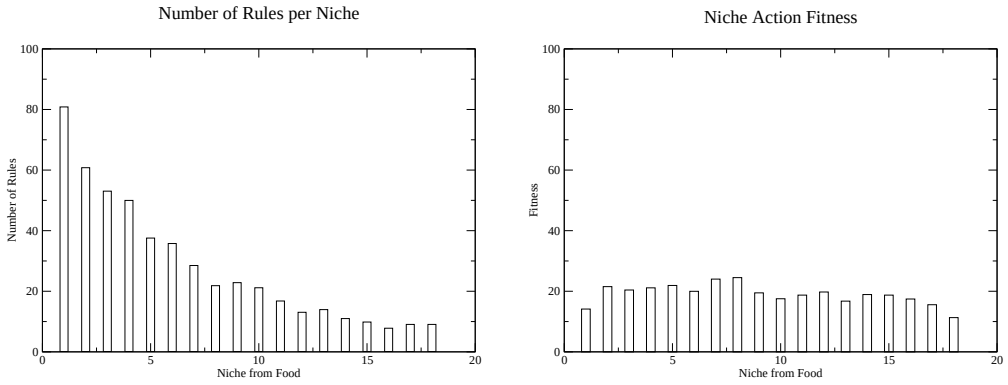
Figure 8(a) shows the performance of ZCS using the parameters given in Section 2 (i.e., those used in Wilson (1994)). It can be seen that ZCS reaches around 3 steps to food. Figure 7(b) shows the performance of ZCS with the same parameters, except $\beta = 0.8$ and $\gamma = 0.02$. Improved ZCS performance in different situations with different parameters from those used by Wilson was shown experimentally in Bull (1998) (see also Tomlinson and Bull (1999a)). In Figure 8(b), we see that ZCS is just about optimal, its performance deviating around 1.9 steps in 10,000 trials.

It is important to note that the GA continues to function and that probabilistic action selection is used, which may account for the deviations. Presentation of results from ZCS's on-line behavior shows near optimal performance. This is in contrast to the usual way in which results are presented for XCS, where only deterministic action selection steps are averaged, also without GA activity (Wilson, 1995). While such a scheme clearly displays the knowledge of the rule base, it is perhaps unrealistic for the practical application of LCS in many cases (e.g., real-time control). However, to indicate the knowledge gained by ZCS, a similar scheme is used for an extra 2000 trials in Figure 8(b). Here the GA is switched off, reinforcement occurs as usual, and an action selection scheme is used that deterministically picks the action with the largest total fitness in [M]. A drop to optimal performance can be seen (Wilson's parameters give a drop to 2.3 steps (not shown)). Indeed, Wilson (1995) proposed ZCS may be able to perform optimally in `Woods1` if it used a similar action selection scheme to that of XCS, but also suggested its "inability to suppress overgenerals, together with the distribution of the prediction over multiple classifiers would still result in a performance and accuracy shortfall versus XCS.". However, as suggested by the models above, the choice of system parameters appears to be at least as important as the action selection scheme. ZCS's sensitivity to the action selection scheme remains open to future investigation.

5.2 `Woods14`

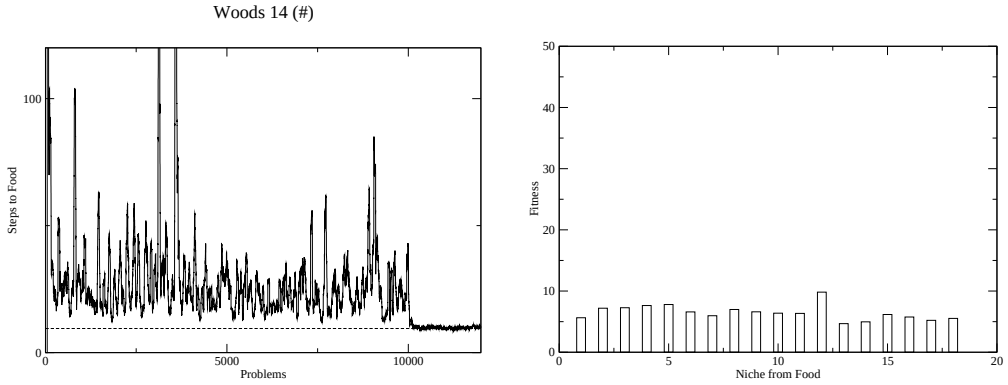
Cliff and Ross (1995) introduced the `Woods14` environment (Figure 7(b)) to examine the performance of ZCS in multistep tasks, where long chains of cooperative rules must emerge. In the full version of `Woods14`, there are 18 blank cells that form a "tunnel" to a food source, optimal behavior being 9.5 steps to food. All other experimental details are the same as those for `Woods1`. Cliff and Ross used the parameters in Wilson (1994), obtaining suboptimal performance (Figure 9(a) (not actually shown in Cliff and Ross (1995)). Their main explanation for this was the effects of overgeneral rules.

Figure 9(b) shows the behavior of ZCS with the same parameters as those in Wilson (1994), except that $P = 800$ and $\beta = 0.8$. We see that ZCS's performance deviates much more than in `Woods1`, achieving around 20 steps to food. Again, an extra 2000 trials were done using the deterministic strategy described above with optimal performance being obtained. Figure 10 shows the load balancing behavior of ZCS after 10,000 trials in `Woods14`, with Figure 10(a) showing how the number of rules per niche decreases the further the niche is from the food source. Cliff and Ross (1995) showed a similar graph. Figure 10(b) shows the fitness of the rules representing the optimal action for each niche, where it can be seen that all fitnesses are roughly equal. That is, ZCS appears to behave as modeled in Sections 3 and 4.



(a) (b)

Figure 10: Analysis of resulting rule bases in `Woods14`.



(a) (b)

Figure 11: Analysis of ZCS started with a greater degree of generalization.

In Section 3.2, the modeled ZCS showed an ability to perform optimally with certain parameters, despite starting with a far greater number of incorrect general rules. Similar experiments have been undertaken using ZCS in `Woods14`. Figure 11(a) shows the performance of ZCS, where the probability of a wildcard under initialization, cover, and mutation was 0.8, as opposed to 0.33. All other parameters remained the same as those in Figure 9(b). It can be seen that there is little change in performance from that seen in Figure 9(b). Figure 11(b) shows the fitness of the rules representing the optimal action for each niche, where it can again be seen that all fitnesses are roughly equal. That is, as seen in Section 3.2, ZCS can give optimal performance in the presence of many overgenerals. Rather, suboptimal performance appears to be caused by a disruption to the fitness sharing process due to inappropriate parameter settings.

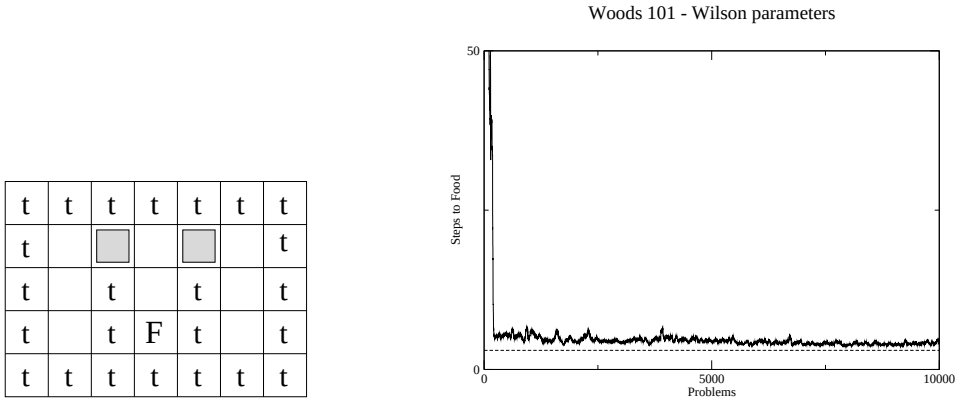
5.3 `Woods101`

Some environments are not completely observable by the learning entity. That is, with respect to the learner's sensory input, environments can be only partially observable, leading to the same sensory input for different environmental states (termed non-Markov). Without internal state, ZCS cannot perform optimally in such cases. Wilson (1994) proposed a simplified version of Holland's (1986) message list: an internal memory register whose state is considered and altered by the rules of the LCS. Cliff and Ross (1994) implemented Wilson's mechanism in ZCS called ZCSM. They found good, but not optimal, performance in the non-Markov mazes `Woods101` and `Woods102`.

The rules of ZCSM are extended to consider not only the external condition (c bits) but also the state of the internal memory register (m bits). Similarly, rule actions are extended to provide the external action (a bits) and an internal action (m bits). Wildcards are allowed for internal actions that leave the state of the corresponding bit in the memory register unchanged, otherwise the register bits are updated to the value defined in the internal action. Hence rules are of the length $c + 2m + a$ bits, and an action set consists of rules proposing the same external and internal actions.

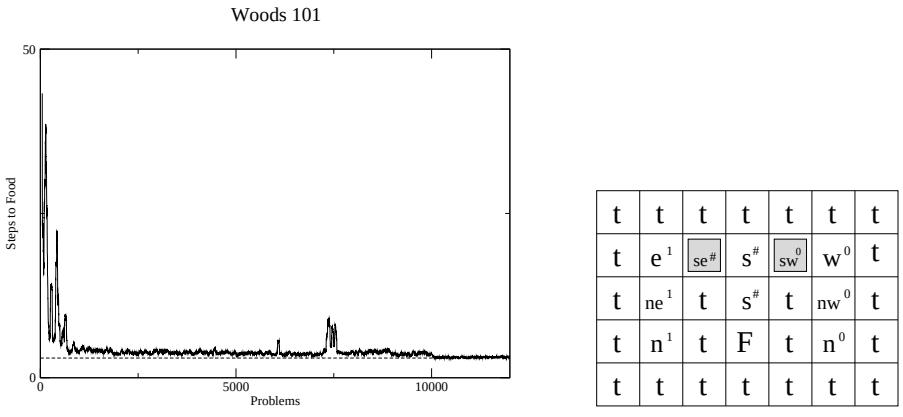
Figure 12(a) shows the non-Markov maze `Woods101`, where optimal performance is 2.9 steps to food. The two marked empty cells show the ambiguous states, i.e., the states under which ZCSM receives the same sensory input but needs to perform a different action for optimal performance. Figure 12(b) shows the non-optimal performance of ZCSM using the parameters given in Wilson (1994) and a memory register of 1 bit (2 states). It should be noted that these runs are done without the additional null external action incorporated by Cliff and Ross. Figure 13(a) shows the performance of ZCSM with $\beta = 0.4, \gamma = 0.4, \mu = 0.01, \rho = 0.45$. We see that its on-line performance is close to optimal and that under the exploit scheme described in Section 5.1, its performance is near optimal (Lanzi and Wilson, 2001) (using Wilson's parameters gives 3.5 steps to food (not shown)). Figure 13(b) shows the external and internal actions of an example solution.

Tomlinson and Bull (1998) examined the performance of ZCSM in a number of increasingly complex mazes. They found that for tasks containing numerous ambiguous states, the memory register must be bigger than expected, i.e., redundancy was required for it to work effectively. Lanzi and Wilson (2001) recently presented optimal and near-optimal performance with XCS incorporating the memory register in a number of non-Markov mazes, including `Woods101`. They also found that for more complex mazes, seemingly superfluous internal memory states were needed. This aspect of its use remains open to further investigation.



(a) (b)

Figure 12: `Woods101` and ZCSM performance.



(a) (b)

Figure 13: Typical strategy learned for `Woods101` and optimal performance from altered parameters.

6 Discussion: ZCS and XCS

On presenting the accuracy-based XCS, Wilson (1995) briefly discussed the role of fitness sharing stating that:

1. it was necessary in payoff-based LCS,
2. it made analysis difficult since rule fitnesses do not clearly relate to external reward,
3. in multistep tasks, early stage setting rules will "due to discounting, appear inherently less fit" and so will be lost,
4. the GA cannot identify overgenerals, and
5. there is no clear tendency to form accurate generalizations.

The last two points are general criticisms of payoff-based fitness and not simply fitness sharing.

The results here would appear to be very much in agreement with the first point, but with the caveat that if fitness sharing is used, suitable system parameters must be identified for a given problem. How difficult it is to tune ZCS in general remains open to further investigation.

Wilson's second point is easily addressed in ZCS if rules maintain a separate "total niche payoff" parameter, updated accordingly, but not used by the system. Since fitnesses generally converge to the same (or very similar) values under fitness sharing, points three and four do not appear relevant to ZCS, although it is noted that the rate of GA activity may be critical.

Regarding the last point: if optimal behavior is required, ZCS seems capable of forming appropriate generalizations, where these will tend to be maximally general with suitable GA activation (Wilson, 1987). XCS builds a full, non-overlapping, maximally general map of the problem space that does have advantages if a posteri explanatory power is required, such as in data mining (see Saxon and Barry (2000) for early use of XCS as a data miner). It is unclear whether this gives XCS a general advantage in nonstationary domains. Hartley (1999) suggested that "if the optimal covering map of knowledge is complete the system has all the rules it needs to solve the problem – it just needs to alter the reward each rule predicts for the performance to be optimal again." However, this is true only if the change does not disrupt any of the generalizations formed by XCS, otherwise it must adjust its rule base. In the cases considered in Section 3.3 with ZCS, switching from the task in Table 1 to that in Table 2 is such a scenario. XCS, assuming that it builds a maximally general, non-overlapping solution, will converge on rules 0:0, #:1, and 1:0 for Table 1. Switching to the task in Table 2, it will need to lose #:1 and evolve the more specific rules 0:1 and 1:1. As was shown above, because of its partial map, ZCS needed only to adjust the proportions from its solution for Table 1 to solve Table 2 (i.e., XCS would have no significant advantage over ZCS). Conversely, in the case of switching from Table 1 to Table 3, XCS would need only to update the predictions for the rules 0:0 and 1:0, leaving the generalization #:1 intact. Typically, this would be a quick process of change. Here the modeled ZCS had to learn a new configuration, which as the dynamics in Figure 5(b) show, was nontrivial (see Hurst and Bull (2001) for use of ZCS, and Hartley (1999) and Hurst and Bull (2002) for use of XCS in nonstationary tasks).

In multistep tasks, the use of accuracy results in XCS being unable to generalize over time steps due to discounting. It also appears to have difficulty with `Woods14`,

although Wilson (2001) identified a number of changes to give optimal performance. Lanzi (1997) has also examined the use of XCS in `Woods14`. He found that by altering the usual 50/50 balance between random explore and deterministic exploit in favor of more exploitation and varying other parameters from the maze task in Wilson (1995), XCS took around 250 steps to reach the food. However, optimal performance was obtained with the introduction of a "specify" operator, which decreased the amount of generalization (number of #'s) of inaccurate rules, in effect pushing XCS towards being a tabular Q-learner (Watkins, 1989). Lanzi suggested the reason why XCS has difficulty in such cases is that, due to a low frequency of visitation for some environmental states, its generalization processes cannot function effectively. A similar point is made by Cliff and Ross (1995) in their discussions of ZCS in `Woods14`. Barry (2001) also examined the performance of XCS in long multistep tasks, showing that it struggles to distinguish between accurate and inaccurate rules further down inductive chains due to the discounting of payoff values. That is, early "stage setting" rules tend to be lost in XCS because the difference in payoffs becomes very small, making the detection of inaccurate rules more difficult. Point three becomes a criticism of XCS. However, Wilson (2000) suggested the evolution of S-expressions, which allow rules to dynamically alter their prediction values depending on the input received. This is currently under investigation by the authors.

Kovacs (2000) compared XCS with a payoff-based LCS. One of the conclusions of his work is that payoff-based LCSs suffer from an "Achilles heel," as they are susceptible to the production of overgeneral rules. However, the comparisons do not consider the use of fitness sharing. The work presented here demonstrates how fitness sharing has the potential to prevent the formation of overgeneral rules and therefore indicates that work comparing the two schemes is far from complete. Indeed, failure in ZCS does not seem to be due to overgeneralization, but to the incorrect setting of system parameters, which prevents the fitness sharing mechanism from working effectively. The debate about the use of payoff-based or accuracy-based LCS has focused almost entirely on the overgeneralization issue. This has been to the detriment of ZCS, even though it prevents overgeneralization through a particularly elegant implementation of fitness sharing. In other areas of machine learning, it is accepted that some algorithms are better suited to certain tasks than others (e.g., neural computation). LCS research should now consider broadening the debate to consider which problem domains are best suited to different forms of the LCS concept. It has already been noted that XCS has an advantage in tasks such as data mining, which need to create and maintain a complete state-action map. ZCS may have an advantage in situations where this map is difficult or impossible to obtain. Hence a rich and multifaceted approach to utilizing LCS in machine learning problems may be envisaged, with payoff-based, accuracy-based, Pittsburgh-style, and heuristic-based (e.g., Stolzmann (1998)) systems all having a role to play. Some systems may indeed have more advantages than others, but this issue remains open to future investigations.

7 Conclusion

In this paper, ZCS has been re-examined. Using simple difference equation models, it has been shown that through its use of fitness sharing ZCS does not necessarily suffer from overgenerals and so has the potential to perform optimally. This general result is important since, as noted in the introduction, most current LCS work has shifted to the use of accuracy-based fitness and a closer alignment to work in reinforcement learning (Wilson, 1995). Significantly, it would appear that the interaction between

the rate of rule updates and the fitness sharing process is critical to whether ZCS can perform optimally. A greater understanding of this aspect of such LCS is now needed and currently under investigation by the authors. It has also been shown that ZCS can be robust to nonstationary problems through its construction of a partial map of the task, depending upon the nature of the change. Based on the results from the models, ZCS has been shown to perform optimally in well-known maze tasks with different parameters from those initially used.

Acknowledgments

Thanks to the members of the Learning Classifier Systems Group at UWE for numerous helpful discussions during this work. Thanks also to Stewart Wilson and the reviewers for their help in improving the original manuscript. The second author is supported by BT Exact and UWE CollR.

References

Ahluwalia, M. and Bull, L. (1999). A Genetic Programming-based Classifier System. In Banzhaf, W. et al., editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18, Morgan Kaufmann, San Francisco, California.

Anderson, H. C. (1991). *An Investigation into Genetic Algorithms and the Relationship between Speciation and the Tracking of Optima in Dynamic Functions*. Honours Thesis, Computer Science Department, Queensland University of Technology, Brisbane, Australia.

Bagnall, A. (2000). *Modelling the UK Market in Electricity Generation with Autonomous Adaptive Agents*. Doctoral Thesis, Computer Science Department, University of East Anglia, Norwich, Norfolk, UK.

Barry, A. (2001). Long Chain Formation in XCS. *Soft Computing*, 6(3). In press.

Bull, L. (1998). On ZCS in Multi-Agent Environments. In Eiben, A. E. et al., editors, *Parallel Problem Solving from Nature - PPSN V*, pages 471–480, Springer Verlag, Berlin, Germany.

Bull, L. (1999). On using ZCS in a Simulated Continuous Double-Auction Market. In Banzhaf, W. et al., editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 83–90, Morgan Kaufmann, San Francisco, California.

Bull, L. (2001a). Lookahead and Latent Learning in ZCS. Technical Report UWELCSG01-004, UWE Learning Classifier Systems Group, University of the West of England, Bristol, UK. Available from http://www.csm.uwe.ac.uk/lcsg

Bull, L. (2001b). On Accuracy-Based Fitness. *Soft Computing*, 6(3). In press.

Bull, L. and Hurst, J. (2000). Self-Adaptive Mutation in ZCS Controllers. In Cagnoni, S. et al., editors, *Real-World Applications of Evolutionary Computing: Proceedings of the EvoNet Workshops - EvoRob 2000*, pages 339–346, Springer, Berlin, Germany.

Bull, L., Hurst, J., and Tomlinson, A. (2000). Self-Adaptive Mutation in Classifier System Controllers. In Meyer, J.-A. et al., editors, *From Animals to Animats 6 - The Sixth International Conference on the Simulation of Adaptive Behaviour*, pages 460–470, MIT Press, Cambridge, Massachusetts.

Cao, Y. J. et al. (1999). Design of a Traffic Junction Controller using a Classifier System and Fuzzy Logic. In Reusch, B., editor, *Proceedings of the Sixth International Conference on Computational Intelligence*, pages 342–351, Springer Verlag, Berlin, Germany.

Cao, Y. J. et al. (2000). Distributed Learning Control of Traffic Signals. In Cagnoni, S. et al., editors, *Real-World Applications of Evolutionary Computing: Proceedings of the EvoNet Workshops - EvoSCONDI 2000*, pages 117–126, Springer, Berlin, Germany.

Cliff, D. and Ross, S. (1995). Adding Temporary Memory to ZCS. *Adaptive Behaviour*, 3:101–150.

Dorigo, M. and Bersini, H. (1994). A Comparison of Q-learning and Classifier Systems. In Cliff, D. et al., editors, *From Animals to Animats 3 - The Third International Conference on the Simulation of Adaptive Behaviour*, pages 248–255, MIT Press, Cambridge, Massachusetts.

Hartley, A. R. (1999). Accuracy-based Fitness Allows Similar Performance to Humans in Static and Dynamic Classification Environments. In Banzhaf, W. et al., editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 266–273, Morgan Kaufmann, San Francisco, California.

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

Holland, J. H. (1976). Adaptation. In Rosen, R. and Snell, F. M., editors, *Progress in Theoretical Biology, 4*. Plenum, New York.

Holland, J. H. (1985). Properties of the Bucket Brigade. In Grefenstette, J. J., editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 1–7, Lawrence Erlbaum, Hillsdale, New Jersey.

Holland, J. H. (1986). Escaping Brittleness. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach, 2*, pages 48–78, Morgan Kauffman, Los Altos, California.

Holland, J. H. and Reitman, J. S. (1978). Cognitive Systems Based on Adaptive Algorithms. In Waterman, D. A. and Hayes-Roth, F., editors, *Pattern-directed Inference Systems*, pages 313–331, Academic Press, New York, New York.

Holland, J. H. et al. (1986). *Induction: Processes of Inference, Learning and Discovery*. MIT Press, Cambridge, Massachusetts.

Horn, J., Goldberg, D. E., and Deb, K. (1994). Implicit Niching in a Learning Classifier System: Nature's Way. *Evolutionary Computation*, 2(1):37–65.

Hurst, J. and Bull, L. (2001). A Self-Adaptive Classifier System. In Lanzi, P. L., Stolzmann, W., and Wilson, S. W., editors, *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, pages 70–79, Springer, Berlin, Germany.

Hurst, J. and Bull, L. (2002). A Self-Adaptive XCS. In Lanzi, P. L., Stolzmann, W., and Wilson, S. W., editors, *Advances in Learning Classifier Systems: Proceedings of the Fourth International Workshop*, Springer, Berlin, Germany.

Kovacs, T. (2000). Strength or Accuracy? A Comparison of Two Approaches to Fitness Calculation in Learning Classifier Systems. In Lanzi, P. L., Stolzmann, W., and Wilson, S. W., editors, *Learning Classifier Systems: From Foundations to Applications*, pages 194–208, Springer, Berlin, Germany.

Lanzi, P. L. (1997). A Model of the Environment to Avoid Local Learning. Technical Report N.97.46 Dipartimento di Elettronica e Informazione, Politecnico di Milano.

Lanzi, P. L. and Wilson, S. W. (2001). Toward Optimal Classifier System Performance in Non-Markov Environments. *Evolutionary Computation*, 8(4):393–418.

Samuel, A. L. (1959). Some Studies in Machine Learning using the Game of Checkers. *IBM Journal of Research and Development*, 3:211–232.

Saxon, S. and Barry, A. (2000). XCS and the Monk's Problems. In Lanzi, P. L., Stolzmann, W., and Wilson, S. W., editors, *Learning Classifier Systems: From Foundations to Applications*, pages 194–208, Springer, Berlin, Germany.

Singh, S. P. et al. (2000). Convergence Results for Single-Step On-Policy Reinforcement Learning Algorithms. *Machine Learning*, 39:287–308.

Smith, S. F. (1980). *A Learning System Based on Genetic Adaptive Algorithms*. Doctoral Dissertation, Computer Science Department, University of Pittsburgh, Pittsburgh, Pennsylvania.

Smith, R. E. and Valenzuela-Rendon, M. (1989). A Study of Rule Set Development in a Learning Classifier System. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 340–346, Morgan Kaufmann, San Mateo, California.

Stolzmann, W. (1998). Anticipatory Classifier Systems. In Koza, J. R., editor, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 658–664, Morgan Kaufmann, San Francisco, California.

Sutton, R. S. (1996). Generalization in Reinforcement Learning: Successful Examples using Sparse Coarse Coding. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M., editors, *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pages 1038–1044, MIT Press, Cambridge, Massachusetts.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning*. MIT Press, Cambridge, Massachusetts.

Tomlinson, A. and Bull, L. (1998). A Corporate Classifier System. In Eiben, A. E. et al., editors, *Parallel Problem Solving from Nature - PPSN V*, pages 550–559, Springer, Berlin, Germany.

Tomlinson, A. and Bull, L. (1999a). On Corporate Classifier Systems: Improving the use of Rule-Linkage. In Banzhaf, W. et al., editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 649–656, Morgan Kaufmann, San Francisco, California.

Tomlinson, A. and Bull, L. (1999b). A Zeroth Level Corporate Classifier System. Technical Report UWELCSG99-001, UWE Learning Classifier Systems Group, University of the West of England, Bristol, UK. Available from http://www.csm.uwe.ac.uk/lcsg

Watkins, C. (1989). *Learning from Delayed Rewards*. Doctoral Dissertation, Computer Science Department, Cambridge University, Cambridge, UK.

Wilson, S. W. (1987). Classifier Systems and the Animat Problem. *Machine Learning*, 2:199–228.

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):149–177.

Wilson, S. W. (2000). State of XCS Classifier System Research. In Lanzi, P. L., Stolzmann, W., and Wilson, S. W., editors, *Learning Classifier Systems: From Foundations to Applications*, pages 194–208, Springer, Berlin, Germany.

Wilson, S. W. (2001). Personal communication.