Adding Temporary Memory to ZCS*

Technical Report CSRP347, February 1995

Dave Cliff and Susi Ross
School of Cognitive and Computing Sciences
University of Sussex, BRIGHTON BN1 9QH, U.K.
Phone +44 1273 678754, Fax +44 1273 671320.
davec@cogs.susx.ac.uk susi@cogs.susx.ac.uk

Abstract

In a recent paper, Wilson (1994b) described a 'zeroth-level' classifier system (ZCS). ZCS employs a reinforcement learning technique comparable to Q-Learning (Watkins, 1989). This paper presents results from the first reconstruction of ZCS. Having replicated Wilson's results, we extend ZCS in a manner suggested by Wilson: the original formulation of ZCS has no memory mechanisms, but Wilson (1994b) suggested how internal 'temporary memory' registers could be added. We show results from adding one-bit and two-bit memory registers to ZCS. Our results demonstrate that ZCS can efficiently exploit memory facilities in non-Markov environments. We also show that the memoryless ZCS can converge on near-optimal *stochastic* solutions in non-Markov environments.

Following the discussion of adding memory, we present results from trials using ZCS in Markov environments requiring increasingly long chains of actions before reward is received. Our results indicate that inaccurate over-general classifiers can interact with the classifier-generation mechanisms to cause catastrophic breakdowns in overall system performance. Basing classifier fitness on accuracy may alleviate this problem. We conclude that the memory mechanism in its current form is unlikely to scale well for situations requiring large amounts of temporary memory. Nevertheless, the ability to find stochastic solutions when there is insufficient memory might offset this problem to some extent.

Keywords:
Classifier Systems; ZCS; Memory; Internal State; Long-Chain Failure; Accuracy.

1 Introduction

In a recent paper, Wilson (1994b) described a 'zeroth-level' classifier system (ZCS). ZCS is strongly inspired by Holland's original "Michigan" classifier system framework (Holland, Holyoak, Nisbett, & Thagard, 1986; Goldberg, 1989; Wilson & Goldberg, 1989), but is intentionally minimalist: several common classifier system features have been stripped away from ZCS in order to clarify the effects of the basic learning mechanisms. In doing so, Wilson was able to show that learning in ZCS bears strong similarities to the widely studied reinforcement-based *Q-Learning* technique (Watkins, 1989; Watkins & Dyan, 1992).

In its original formulation, ZCS has no memory mechanisms: the input-output mappings learned by ZCS are therefore always purely *reactive*, in that the current output of the system is determined solely by the current input. Wilson recognized this limitation, and proposed a method by which extra internal state, conceptualized as the system's "temporary memory",

*Submitted to *Adaptive Behavior*.

could be added to ZCS. Wilson's proposal involved adding an internal memory register consisting of a few binary digits (bits), and augmenting the system's set of possible actions to include actions which set or reset individual bits in the register. Over time, ZCS should adapt to manipulate and exploit internal state in situations where this is appropriate.

In this paper we present results from experiments with ZCS extended to incorporate Wilson's proposed memory system: we refer to the extended system as ZCSM. Our results demonstrate that Wilson's proposals do work, at least for small ($b < 3$) b-bit internal registers.

For completeness, Section 2.1 gives a summary of ZCS, followed by our replication of Wilson's published results in Section 2.2. Section 3 presents our results from extending ZCS to incorporate temporary memory. Results from testing ZCSM in a variety of environments indicated that occasionally an adapted population of classifiers could exhibit severe failures in performance which were both sudden and unpredictable. To explore the cause of these failures, Section 4 describes results from extended testing of ZCS in situations where increasingly long chains of sequential actions are required to reach the reward state: we found that ZCS grew less stable as the chain-length increased. We argue that this is due to two factors: greedy classifier creation and conflicting over-general classifiers. The second factor could be alleviated by basing the fitness of classifiers on their accuracy.

To demonstrate that the effects we identify are not artifacts of the spatially discrete 'grid-world' environments employed in our experiments, Section 5 presents results from using ZCS for animat guidance in environments where the animat's spatial location is continuous and its sensory input is more firmly analogous to visual sensing: we show that similar stability problems occur. Conclusions are drawn in Section 6.

2 ZCS: Overview and Replication

2.1 Overview

The definitive paper on ZCS is (Wilson, 1994b), to which we refer the reader for full details. For the sake of completeness, we include an overview here.

ZCS is viewed as a mechanism which interacts with some environment: detectors supply a binary encoded *sense-vector* which affects the *action* chosen by ZCS: these actions are executed by *effectors* which may alter the state of the environment in some manner.

ZCS maintains a single population, [P], of N condition-action classifiers. The conditions are encoded using the ternary alphabet $\{0, 1, \#\}$ where 0 and 1 are used to match against sensory input, and # acts as a "don't-care" wild-card, allowing generalization. Actions are also encoded using a discrete alphabet. The conditions and actions in the initial population are set randomly: the probability distribution of characters c (i.e. 'alleles') at each locus in the condition is given by: $\Pr(c = \text{'\#'}) = P_{\#}; \Pr(c = \text{'0'}) = \Pr(c = \text{'1'}) = 0.5(1.0 - P_{\#})$. Each classifier has an associated scalar strength, set initially to a value S_0 for all classifiers.

ZCS operates iteratively in discrete time-steps. On each time-step, the system passes through stages of *performance*, *reinforcement*, and *discovery*.

In the performance stage, the current sensory input is compared with the conditions of all classifiers in [P]: a condition matches the input if there is a 1 in the input at each locus where there is a 1 in the condition, and likewise for 0's; a wild-card # in the classifier condition matches either a 0 or a 1 in the input. All matching classifiers become members of the *match-set* [M]. If ever there are no matching classifiers in [P] (i.e. if [M] is empty), or if the total strength of the classifiers in [M] is less than a fixed fraction ϕ of the mean strength of classifiers in [P], a *covering* process is invoked: a classifier is selected for deletion (using roulette selection on reciprocal of strength) and is replaced by a classifier whose condition is a copy of the current sense vector with characters at each locus changed to # with probability $P_{\#}$. The action for this classifier is chosen at random using a uniform distribution over the space of possible actions,

and its strength is set to the population average.

The members of [M] will often have different actions: [M] is partitioned into a number of potential *action-sets*, where each member of a given action-set advocates the same action. The strengths of the classifiers in each action-set are summed, and one action-set is chosen using roulette-selection on these total strengths: this action-set is referred to as [A]. The potential action-set with the highest total strength (which we refer to as the *max-set* [H]) is also retained for use in the reinforcement stage. The action a advocated by the classifiers in [A] is sent to the system's effectors for execution. Depending on environmental circumstances, a scalar reward reinforcement value r_{imm} may be supplied to ZCS as a consequence of executing a.

In the reinforcement stage, a "Bucket-Brigade" credit-assignment policy similar to Q-Learning is employed: each member of [A] has a fixed fraction $\beta : \beta \in (0, 1] \subset \mathbf{R}$ deducted from its strength (β modulates the 'learning rate' of the system). Next, each classifier in [A] has its strength increased by $\beta r/|A|$ where $|A|$ is the number of classifiers in [A] and (in the work reported here) $r \in \{0, r_{\text{imm}}\} : r_{\text{imm}} > 0$ (i.e. the reinforcement value is either zero or a fixed positive value). After this, the classifiers in the *previous* action set (denoted $[\text{A}]_{-1}$) have their strengths incremented by a value $\beta\gamma \text{S}_{[\text{H}]}/|\text{A}_{-1}|$ where: γ is a discount factor $\gamma \in (0, 1] \subset \mathbf{R}$; $\text{S}_{[\text{H}]}$ is the total strength of classifiers in [H]; and $|\text{A}_{-1}|$ is the number of classifiers in $[\text{A}]_{-1}$. Finally, all classifiers in the set difference [M]–[A] have their strengths reduced by a small fraction τ, which acts as a 'tax' to encourage exploitation of strong classifier sets. The similarities between this learning regime and Q-Learning are described in detail by Wilson (1994b); Dorigo and Bersini (1994) have also discussed the relationship between classifier systems and Q-Learning.

In the discovery stage, classifiers are generated via a panmictic genetic algorithm (GA). On each iteration, there is a probability ρ that the GA is invoked:[1] if it is, two classifiers are roulette-selected on the basis of their strengths; these are copied with mutation and/or crossover to form two new classifiers. Mutation operates with probability μ at each locus during copying: if a mutation occurs, the character (allele) at that locus is mutated equiprobably into one of the other allowed alleles. Crossover operates with probability χ: if it occurs, a single crossover point is chosen at random with uniform probability over the classifier loci. Half of the strength of each parent classifier is deducted in discovery and assigned to its copy: if crossover occurs then the strengths of the offspring are set to the mean of the two inherited strengths. The offspring replace two classifiers chosen for deletion at random, using roulette selection on the reciprocals of the strengths of the classifiers in [P].

Following completion of discovery, the system returns to the performance stage: the execution of action a may have altered the state of the environment sufficiently to change the sense-vector. The iteration through these stages continues until the system reaches some halting condition, such as having received a predefined number of rewards.

Although ZCS is sufficiently general to be applied in a wide variety of domains, in this paper we concentrate on applying ZCS to the problem of guiding an idealized autonomous mobile agent through an environment which is initially unknown. Results for such problems were also presented by Wilson (1994b); we describe the environments and show replication of Wilson's results in the following section.

[1] Wilson (1994b, p.5) formally defines ρ as the "average number of new classifiers generated by the GA per time-step...". Strictly, our implementation of ρ differs: on each pass through the discovery stage, there is a probability ρ that the GA will be invoked *once*. As a single invocation of the GA creates two new classifiers, the average number of new classifiers generated by the GA per time-step is 2ρ in our implementation. Furthermore, in our implementation ρ only takes meaningful values in the range $[0, 1]$, whereas under Wilson's definition values of $\rho > 1$ would be possible. However, Wilson's published results are with $\rho = 0.25$; we therefore set $\rho = 0.125$ in our system, yielding the same net effect.

2.2 Replication of Wilson's Results

Wilson (1994b) gives results of ZCS operating in 'woods'-style environments (cf. Wilson, 1985; Cliff & Bullock, 1993). In both, the classifier system is considered as being responsible for guiding an 'animat' (i.e. an artificial creature: (Wilson, 1985, 1987, 1991)) through environments composed of two-dimensional grids of cells. Cells may be occupied by an obstacle (represented as T for 'tree'), by a reward (represented as F for 'food'), or may be blank. The animat can occupy blank cells, and can move into F reward-cells but not into T obstacle-cells. The animat moves in discrete steps: on each step it may move into one of the eight surrounding cells. In all the environments discussed in this paper, toroidal wrap-around occurs if the animat attempts to move off an edge. At each cell, the animat 'senses' the contents of the eight surrounding cells, encoding the contents of each using a 2-bit binary code (i.e.: 11=F; 10=T; 00=blank) to form a 16-bit *sense-vector* against which classifier conditions can be matched.

A single 'run' of the animat adapting to its environment consists of a set number of *trials*. Before the first trial in each run, the classifier population is randomly initialized. At the start of each trial, the animat is positioned at a randomly chosen blank cell. The animat then makes a number of steps: on each step, the ZCS selects a movement action, then attempts to execute that action (attempting to move onto a T results in the animat not moving at all). If the animat moves onto a F cell, r_{imm} is supplied to ZCS, and the trial ends. Given such an experimental regime, the 'aim' of the animat is to move onto an F cell in as few steps as possible (attempting to move onto a T counts as a step, despite the lack of movement). To monitor improvements in performance, a running average of the number of steps to food over the previous 50 trials is calculated at the end of each trial: this performance measure is referred to as `stpsav`.

Wilson demonstrated the progress of ZCS-guided animats in two environments, referred to as woods1 and woods7: figure 1 shows woods1. Woods1 is a *Markov* environment for the animat: the environment is such that the values in the animat's sense vector are, at any time, sufficient to determine the global state of the environment (i.e. the animat's position), and the effects of the animat's action depends only on the chosen action and the current global state of the environment; hence the current values in the sense-vector should always be sufficient in principle to determine an action which takes the animat nearer to the F.[2]

Results from runs using ZCS with the animat in woods1 are shown in Figure 2.[3] As can be seen, the average value of `stpsav` very rapidly falls to near-optimal levels. Figure 3 shows a 'vector-field' representation of the movements exhibited by the adapted ZCS classifier population: because the action-set is chosen stochastically from the match-set, the animat's movement policies are non-deterministic, so the vectors indicate a probability distribution over the possible actions at each cell.

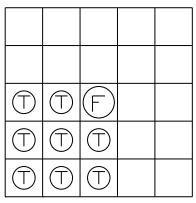

Figure 1: The environment woods1. Trees denoted by T; Food (reward) by F.

A more challenging environment, woods7, is illustrated in Figure 4. Woods7 is *non-Markov* in that current sensory input is *not* always sufficient to uniquely determine a move which takes

[2]It is important to note that it is the *interaction* between the environment and the agent's sensory sampling strategy (cf. Cliff & Bullock, 1993) which determines whether the environment is Markov or not.

[3]Copies of the code, in both C and POP-11, are given in (Ross, 1994).

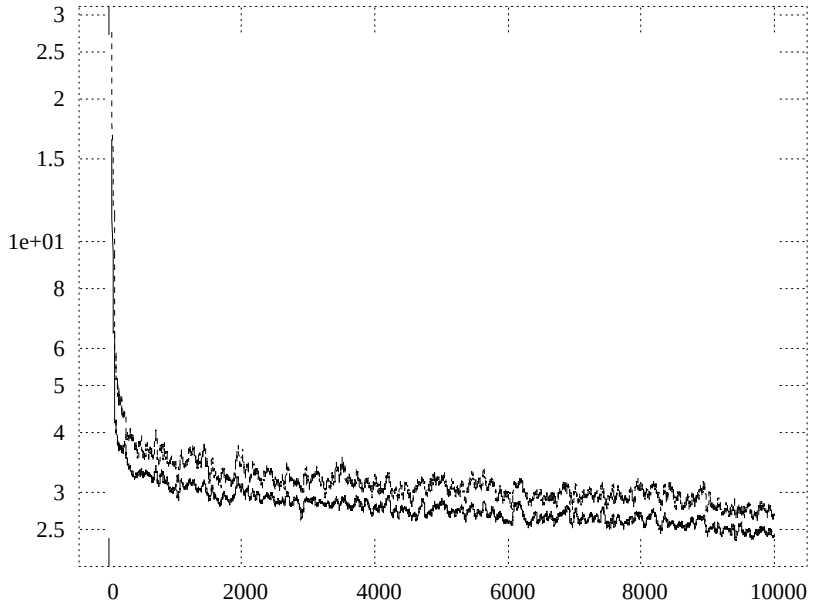

Figure 2: ZCS in woods1. Average of 10 runs. Ordinate is trial number. Abscissa: solid line is mean stpsav; dashed line is mean plus one standard deviation. Parameter values as for (Wilson, 1994b), i.e.: $N = 400$, $P_{\#} = 0.33$, $S_0 = 20.0$, $\beta = 0.2$, $\gamma = 0.71$, $\tau = 0.1$, $\chi = 0.5$, $\mu = 0.002$, $\rho = 0.25$, $\phi = 0.5$, $r_{\mathrm{imm}} = 1000$.

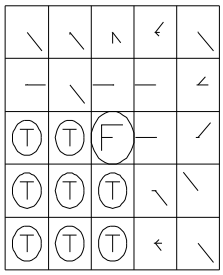

Figure 3: Vector field for ZCS adapted to woods1, after 10000 trials. Each blank cell has up to eight vectors indicating the probability that the animat will move in each of the eight possible directions: the longer the vector, the nearer the probability is to unity; normalized so maximum vector length is equal to half the side-length of a cell.

the animat in the direction of the nearest F. This is because many of the cells in the environment are *perceptually aliased* (cf. Whitehead & Ballard, 1990): the same sensory input is received at more than one position in the environment, and hence sensory input alone cannot determine the global state. In particular, there are large numbers of cells in woods7 where the animat is surrounded by blank cells, so all the bits in the sense vector are zero: in these perceptually aliased cells, there is insufficient information to determine an appropriate action (i.e. one which moves the animat towards the nearest F). Results for using ZCS in woods7 are shown in Figure 5. Again, the system rapidly adapts to generate near-optimal movement patterns, but perceptual aliasing helps prevent truly optimal movement policies from being discovered. Further discussion of reinforcement learning in non-Markov environments can be found in (Whitehead & Lin, 1993).

The results in Figures 2 and 5 are in close agreement with Wilson's results for using ZCS in

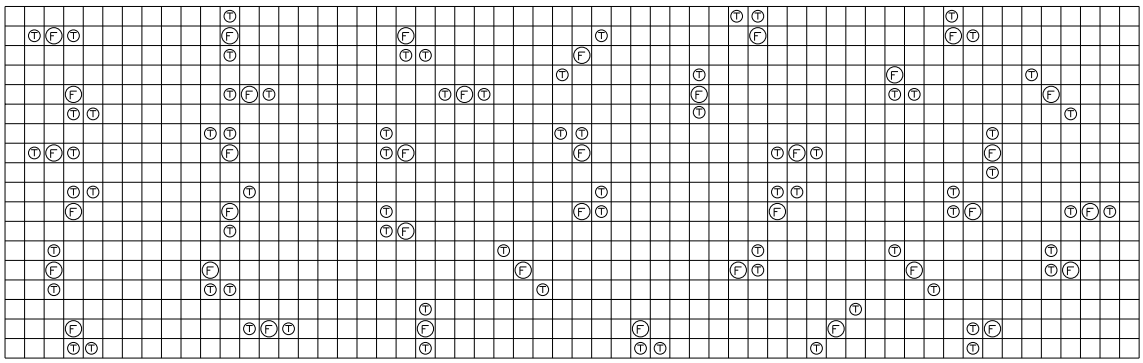

Figure 4: The environment woods7.

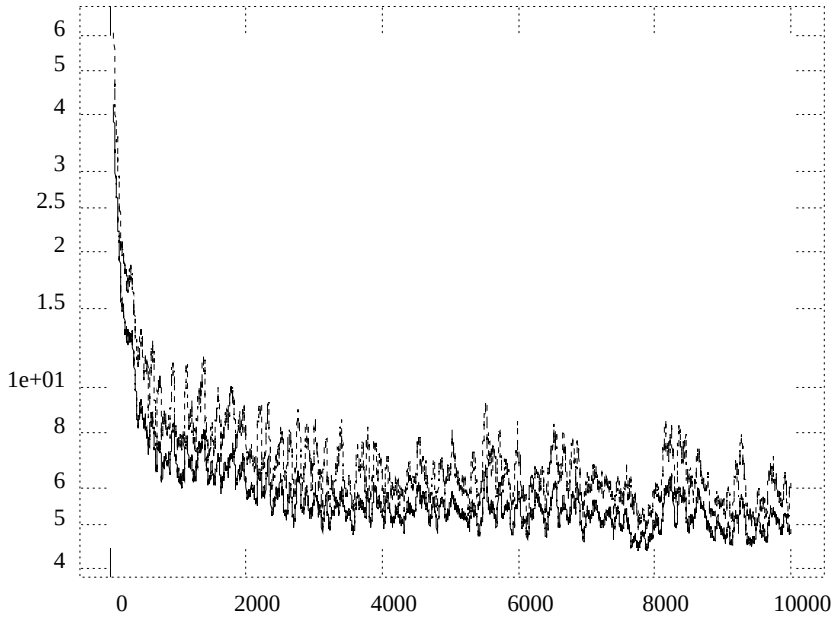

Figure 5: ZCS in woods7. Average of 10 runs. Format and parameters as for Figure 2.

the same environments (Wilson, 1994b, Fig.3 p.7 and Fig.6 p.9), with the same ZCS parameter values (Wilson, 1994b, p.7). Thus, to the best of our knowledge, our data presented here constitute the first replication of Wilson's published results.

3 Adding Memory

3.1 Wilson's Proposal

As was noted in Section 1, the absence of internal state in ZCS restricts it to learning only *reactive* input-output mappings, where the current output of the system is determined solely by the current input. Wilson (1994b, pp.11–12) noted this limitation and proposed an extension to ZCS where the system is given an internal b-bit 'memory register', and the set of possible actions is extended to include operations which may set or reset individual bits in the register. These *internal* actions could be performed in parallel with *external* actions (in the animat context, external actions are movements in the environment). The system is also given a *null* external action: in principle this allows for the execution of sequences of internal actions without performing any external actions. (Metaphorically, this can be viewed as giving the system the

opportunity to "sit and think"). Wilson describes this extension as the addition of "temporary memory": for consistency, we will use the same terminology, although we acknowledge (and agree with) the comments of Colombetti and Dorigo (1994, pp.250-251), who note that terms such as *memory* and *representation* are often deceptively subjective labels referring to the agent's internal state, and that internal state is a more neutral term.

We refer to ZCS with b bits of memory as ZCSMb: when discussion of ZCSMb is independent of the value of b, we refer simply to ZCSM.

3.2 Implementation

ZCSM was implemented in the manner proposed by Wilson. The classifier condition is divided into two sections: the first is used to match against the sense-vector, as in ZCS; the second is a sequence of b characters from the ternary alphabet $\{0, 1, \#\}$ which is matched against the current settings of the b bits in the memory register. The classifier action is also in two parts: the first part specifies one of the 9 possible external actions (eight movements and one null), encoded as two characters from $\{0, 1, \#\}$. The second part is the internal action, being a sequence of b characters from the same ternary alphabet: a 0 or 1 in the internal action specifies that the corresponding bit in the register should be set to that value; a # indicates that the corresponding bit of the register should be left unaltered.

In the formation of the initial random population of classifiers in ZCSM, characters for both conditions and both actions are generated at random, using the same probability distribution as for the generation of the sensory condition in ZCS. At the start of each trial, the register is initialized by setting all bits to zero. In breeding, crossover can occur at any point in the string formed by concatenating the characters representing the sense-condition, memory-condition, internal-action, and external-action; in that order.[4]

Adding the internal register does not require the introduction of any new parameters. In the experiments reported below, all parameter values were the same as those used in the ZCS replication experiments described previously in Section 2.2.

3.3 Results

Although internal state can be an advantage in a variety of situations, our primary interest was to test the ability of ZCSM in dealing with non-Markov woods-style environments. Depending on the structure of the environment, ZCSM should be able to employ the memory mechanisms to disambiguate perceptually aliased global states. That is: if a given sense vector can correspond to one of several global positions with conflicting appropriate actions, it is possible in certain environments for the internal state to be manipulated in such a way that the memory register augments the (aliased) sensory input so that an appropriate action can be selected. This is well illustrated by considering the environment woods101 shown in Figure 6.[5]

There are two distinct cells in woods101 which generate the same sense vector (see Figure 7), but have *opposite* appropriate actions. In principle, a one-bit memory is sufficient to disambiguate these aliased states. For example: the system could adapt so that whenever it is in any of the unaliased cells on the west side of woods101 it sets the register to 1, while at all other cells it leaves the register alone (recall that all the bits in the memory register are initialized to 0). If the animat's input is the sense-vector for the aliased cells, then the correct action is to move south-east if the value in the register is 1; otherwise move south-west and set the register to 1.

[4] This is a slight difference from Wilson's proposal, where the order was sense-condition, memory-condition, external-action, and internal-action. We reason that placing the memory-condition and internal-action closer together at crossover reduces the chances of good internal condition-action pairs being separated in breeding.

[5] This environment is perhaps better known as "McCallum's maze" (Littman, 1994, p.243). We refer to it as woods101 for consistency with the woods102 environment introduced later.

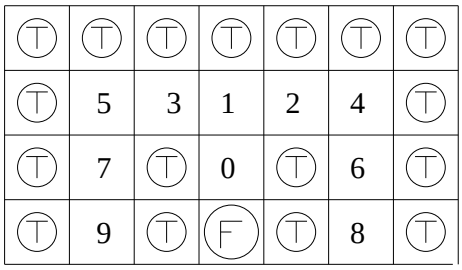

Figure 6: The environment woods101. The numbers in the cells are labels used in analysis (see Figure 10).

```
T  T  T
.  *  .
.  T  .
```

Figure 7: Sensory alias in woods101: there are two cells (labeled 2 and 3 in Figure6) where the animat (*) samples a sense-vector illustrated here by the pattern of neighbouring cells (blank cells represented by '.'). In one, the correct action is to move westward; in the other, the correct action is to move eastward.

Results for ZCSM1 in woods101 are shown in Figure 8: as can be seen, it learns to move in a near-optimal fashion. Figure 9 shows the vector field for the adapted system: the population of classifiers is using the memory register in a manner similar to that outlined above. If the animat starts a trial in one of the two cells on the vertical center-line of woods101 then it moves directly toward the F (because the register is always initialized to zero). If it starts at any cell to the left of the center line, then it can only enter the left-hand aliased cell with memory=0, which moves the animat onto the cell immediately adjacent to the F. If the animat starts in any cell to the right of the center line, it will switch state to memory=1 in either one or two steps, landing on the right-hand aliased cell with memory=1. This also results in the animat moving onto the cell immediately adjacent to the F. Figure 10 summarises the most likely trajectories resulting from this vector field. As can be seen, the memory is being used to avoid the problem of aliased cells with conflicting appropriate actions.

As a control, we ran experiments with ZCSM0 in woods101. Our expectation was that the absence of memory would prevent the co-adaptation of effective sets of classifiers. Results are shown in Figure 11. As can be seen, the values for stpsav at each trial are generally higher with ZCSM0 than with ZCSM1 (cf. Figure 8), but they are nevertheless significantly lower than values expected from a random walk strategy. The vector field for a ZCSM0 classifier population adapted to woods101 is shown in Figure 12: the significant point to note in the vector field is that the (memoryless) ZCSM0 has adapted to a state where the aliased cells have two actions advocated with almost equal probability; in both aliased cells, one of the actions takes the animat to a cell where the F falls within the sense vector, and the other action has the effect of moving the animat to a cell where the sole action is to *return* the animat to the aliased cell. Hence, the two actions at the aliased cells correspond to an efficient *stochastic* solution to the problem of insufficient sensory information: figuratively, we can think of the animat as "flipping a coin" to decide between the two alternative actions when faced with aliased sensory input; with no memory, it can do no better given that the uniformly distributed starting positions lead to it visiting both aliased cells with equal probability.

To test the capabilities of ZCSM2, we designed a new environment: woods102, illustrated

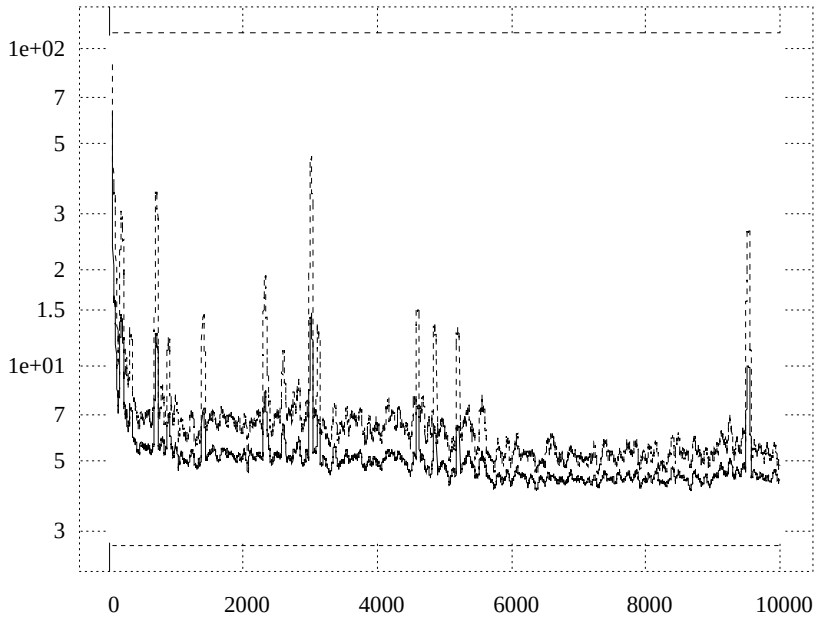

Figure 8: Results for ZCSM1 in woods101. Average of 10 runs. Lower horizontal dashed line shows theoretical optimum performance; upper horizontal dashed line shows mean performance using random walk.

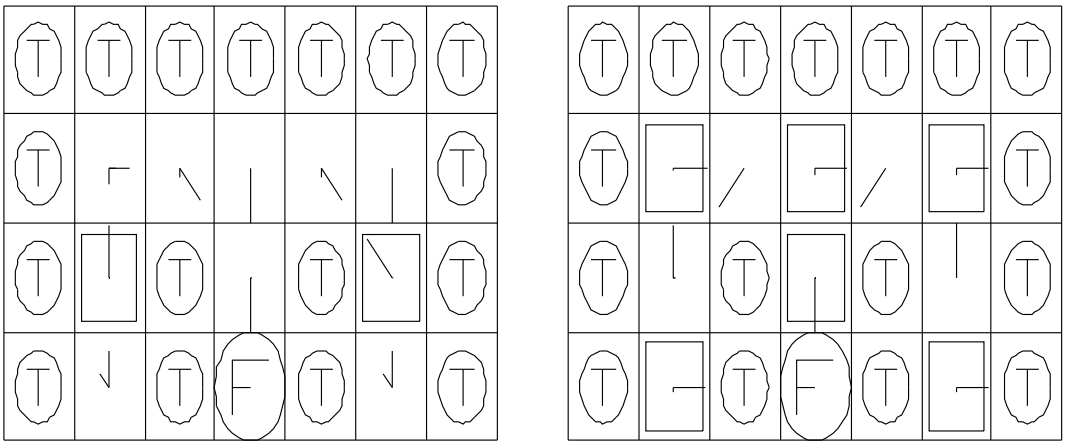

Figure 9: Vector field resulting from ZCSM1 adaptation in woods101 after 10000 trials. Display format as for Figure 3, extended to indicate manipulation of internal state. Left-hand figure is vector field when memory=0; right-hand figure is vector field when memory=1. Cells with a double-border indicate that the most likely internal action in that cell involves flipping the value of the memory-bit, while performing the external action indicated.

Starting Cell	Most Likely Trajectory
0_0	F
1_0	0_0 F
2_0	6_0 2_1 0_1 F
3_0	0_0 F
4_0	6_0 2_1 0_1 F
5_0	3_0 0_0 F
6_0	2_1 0_1 F
7_0	5_1 3_0 0_0 F
8_0	6_0 2_1 0_1 F
9_0	7_0 5_1 3_0 0_0 F

Figure 10: Most likely trajectories resulting from the vector field shown in Figure 9. Each line in the table shows a starting cell, followed by the most likely trajectory from that cell through woods101 to the F. The cell labels are those introduced in Figure 6. The subscript to each label shows the contents of the memory register. Note that the memory register is set to 0 at the start of each trial.

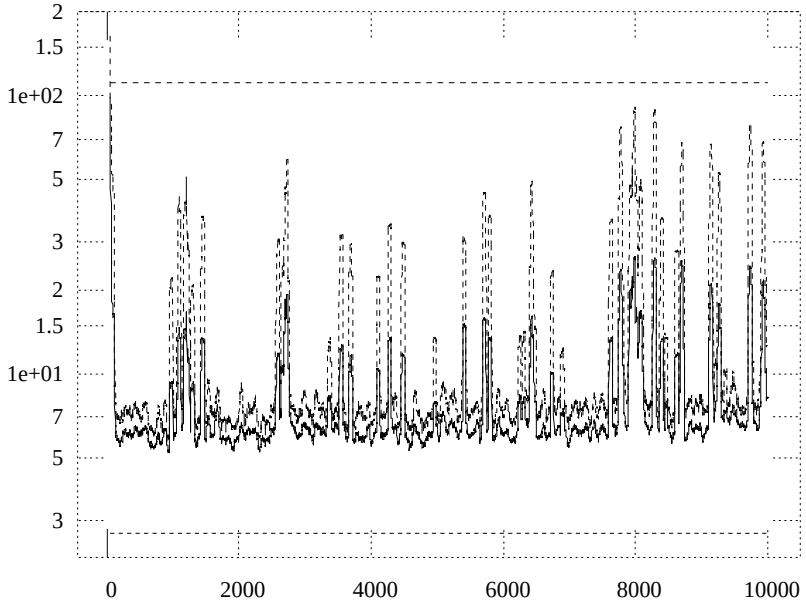

Figure 11: Results from ZCSM0 in woods101. Average of 10 runs. Format as for Figure 8.

in Figure 13. This environment is more challenging than woods101 in two respects: first, there is a sense-vector which acts as an alias for four different cells, all of which require a different appropriate action; second, there is another sense-vector which acts as an alias for two other cells, which again require different actions: these aliased sense-vectors are illustrated in Figure 14.

Given that one of the sensory aliases in woods102 can be generated when the animat is at one of four different positions in the world, in principle two bits of memory (giving $2^2 = 4$ distinct internal states) should be sufficient to disambiguate the aliases. Results for ZCSM2 in woods102 are shown in Figure 15. Again, with sufficient memory ZCSM is capable of learning efficient sets of classifiers. As with woods101, we ran control experiments using ZCSM with insufficient memory: Figure 16 illustrates average performance for ZCSM1, and Figure 17 shows a vector

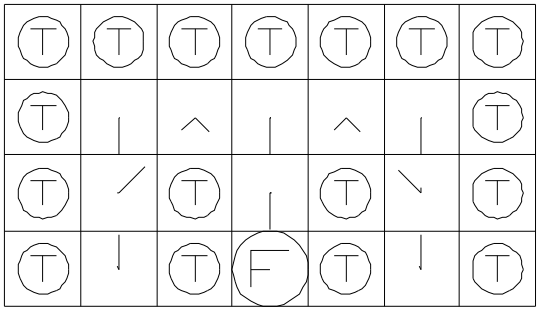

Figure 12: Vector field from ZCSM0 in woods101, after 10000 trials.

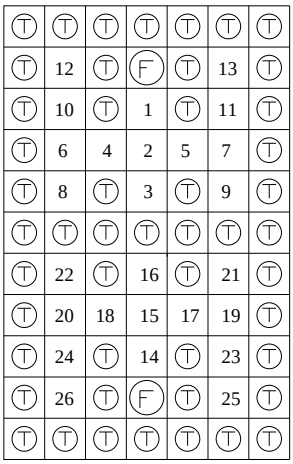

Figure 13: The environment woods102. The numbers in the cells are labels used in analysis.

```
. T .          T . T
. * .          . * .
. T .          T . T
```

Figure 14: Aliased sensory input in woods102: the input pattern on the left occurs in four cells (labels 4, 5, 17, and 18); the input pattern on the right occurs in two cells (labels 2 and 15). For both patterns, different appropriate actions are required at the different cells.

field from a ZCSM1 classifier population adapted to woods102. Once again, in the absence of sufficient temporary memory, the system has converged on a solution which, although giving worse performance than that obtained with sufficient memory, is nevertheless an effective solution to the problem of aliased sensory states. In the case of the vector field shown in Figure 17, the adapted solution does not depend on stochastic effects at the aliased cells. Rather, long circuitous routes are taken through woods102: some examples of these are listed in Figure 18. As can be seen, the circuitous routes avoid the problems of sensory aliasing at cells 4, 5, 17, and 18 by ensuring that, if the trial starts in the southern half of woods102 (cells 14 to 26) then the F is approached from the east (i.e. via cell 17 with memory=1), regardless of whether the trial started in the east or west of the environment. Likewise, in the northern half (cells 1 to 13) the F is always approached from the west (i.e. via cell 4 with memory=0), again regardless of where

the trial started: the circuitous routes ensure that the memory bit is set in such a way that when the aliased cell is reached, an appropriate action is generated. As can also be seen from the trajectories in Figure 18, the most likely trajectories from the aliased cells on the vertical center-line (i.e. cells 2 and 15) use the memory register in a similar manner.

It should be noted that the performance graphs for ZCSM in both woods101 and woods102 are less smooth than those for woods1 and woods7: even after several thousand trials, the average values of `stpsav` can return to high levels close to those exhibited in the early trials (i.e. before adaptation has converged on a good set of co-adapted classifiers). The nature of this apparent loss of stability is discussed further in Section 4.

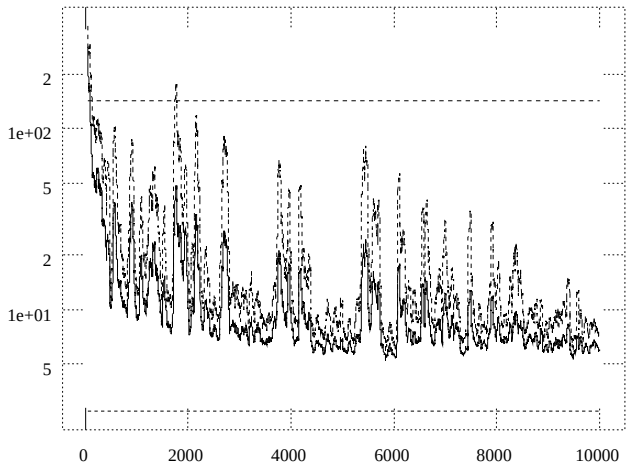

Figure 15: ZCSM2 in woods102. Average of 10 runs. Format as for Figure 8.

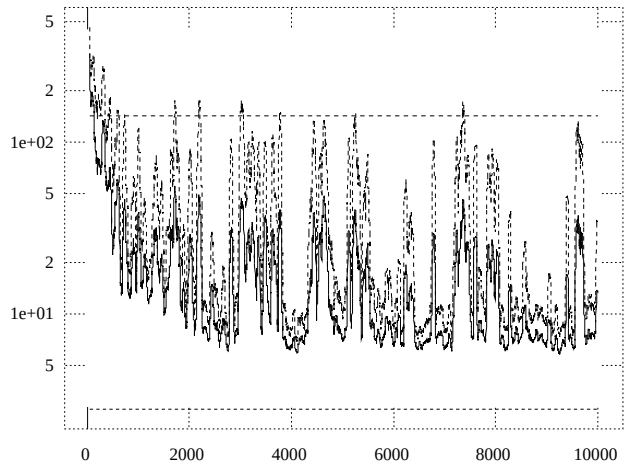

Figure 16: ZCSM1 in woods102. Average of 10 runs. Format as for Figure 8.

Figure 19 shows the progress of a single run with the memoryless ZCSM0 in woods102. As can be seen, there are extended periods where `stpsav` is close to the theoretical optimum value: during these periods a co-adapted classifier population has developed a stochastic solution to the problem of aliased states. Figure 20 shows the vector field during one of the periods of good performance (at trial 800 in the run shown in Figure 19): in almost all unaliased cells a single appropriate action is advocated; in the four aliased cells there are two actions of similar strengths, corresponding to a stochastic solution to the problem. As is also clear from Figure 19,

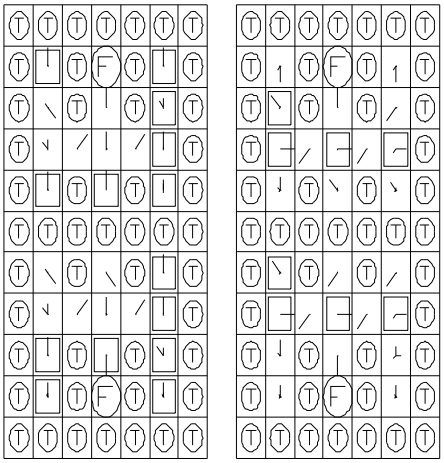

Figure 17: Vector field resulting from ZCSM1 adaptation in woods102, after 10000 trials: format as for Figure 9.

Starting Cell	Most Likely Trajectory									
2_0	1_0	F								
12_0	12_1	10_1	10_0	4_0	1_0	F				
13_0	13_1	11_1	5_1	3_1	4_1	8_1	6_1	4_0	1_0	F
15_0	16_0	17_0	21_0	21_1	17_1	14_1	F			
25_0	23_1	19_1	19_0	21_0	21_1	17_1	14_1	F		
26_0	24_1	20_1	18_0	16_0	17_0	21_0	21_1	17_1	14_1	F

Figure 18: Most likely trajectories resulting from the vector field shown in Figure 17. Each line in the table shows a starting cell, followed by the most likely trajectory from that cell through woods102 to an F. The cell labels are those introduced in Figure 13. The subscript to each label shows the contents of the memory register. Note that the memory register is set to 0 at the start of each trial. See text for discussion.

the stochastic solutions are unstable: occasionally, the value of `stpsav` rapidly rises to values close to those given by the system before adaptation has taken place. Figure 21 shows the vector field during one of these periods of failure (at trial 10000 in the run shown in Figure 19): as can be seen, the previously well-adapted classifier population has been almost entirely disrupted, leading to a failure in system performance. The nature of these failures is explored in detail in Section 4.

3.4 Discussion

The results presented in this section have demonstrated two main points: first, that ZCSM as proposed by Wilson can exploit available internal state to disambiguate aliased sensory inputs; second, if there is insufficient internal state-space, ZCSM can converge on efficient stochastic solutions.

Although our implementation demonstrates that Wilson's proposed memory mechanism does indeed work, we have doubts about its extensibility. The combinatorics of implementing Wilson's proposal are unappealing: the number of possible internal actions rises exponentially with the number of bits of temporary memory (for b bits of memory, there are 3^b possible internal actions).

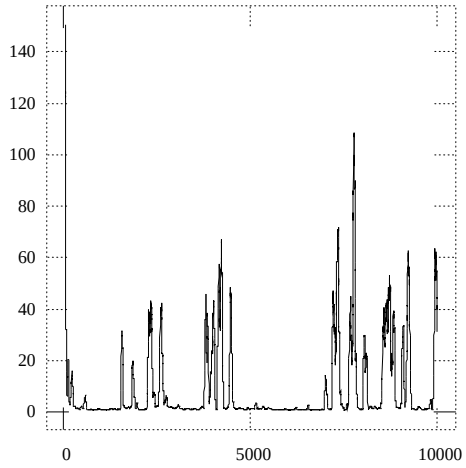

Figure 19: ZCSM0 in woods102: stpsav over one run.

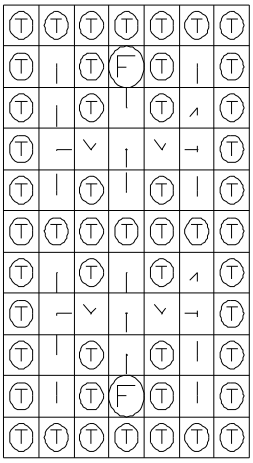

Figure 20: ZCSM0 vector field at trial 800 from the run shown in Figure 19: see text for discussion.

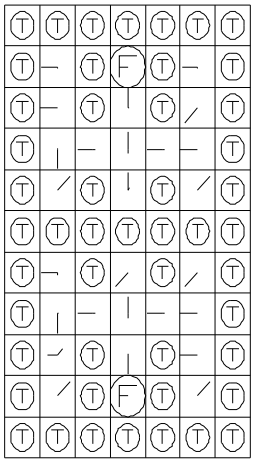

Figure 21: ZCSM0 vector field at trial 10000 from the run shown in Figure 19: see text for discussion.

For the purposes of discussion, let n_a denote the total number of actions (internal-external combinations, including null actions) that it is possible to encode on a ZCSM classifier. Because each classifier can advocate both an internal action and an external action, the animat with 9 possible external actions and b bits of memory has $n_a = 3^{2+b}$. We believe that adaptation is likely to be significantly retarded if n_a is appreciably higher than the population size N. In all the experiments reported above, we have used $N = 400$, and the highest number of possible actions that we have worked with is $n_a = 3^{2+2} = 81$ in ZCSM2. Hence in our experiments the initial random population of classifiers should contain, on the average, about five classifiers per possible internal-external action combination in ZCSM2, about 15 in ZCSM1, and about 44 in ZCSM0. The key issue here is that, on the average, the space of possible actions is densely covered by the initial classifier population.

Now consider ZCSM8: eight bits of memory is not an outrageously large internal state-space, but in ZCSM8 $n_a = 3^{2+8} = 59049$; if N is kept at 400, then no more than 0.68% of the possible internal-external action combinations can be represented in the initial classifier population! Hence the space of possible actions is only very sparsely populated by the initial classifiers, and

so the system will have to rely on crossover and mutation to generate action combinations not represented in the initial population, with a corresponding reduction in the rate of adaptation. If the population size is increased in line with n_a then the problem should be alleviated, but at the cost of increased storage expense and longer search times when forming the match-set. Also, partial remedy may be offered by restricting the space of possible actions encoded on the classifiers to just the union set of external actions and internal actions, so the action advocated by each classifier is either internal or external but never both, giving $n_a = n_e + 3^b$ for a system with n_e external actions.

Further efficiency problems may be caused by the relatively unsophisticated memory manipulation facilities in ZCSM as presented here: although in principle the set/reset/leave-alone operators at each bit position may be sufficient to effect sophisticated internal processing capabilities, it would require long chains of internal actions to achieve either simple Boolean logic operations or simple binary math capabilities; in Section 4 we question the ability of ZCS to form and maintain such long chains of actions. For this reason, a more profitable path to follow may be extending ZCS to employ memory-manipulating *s-classifiers* (where the classifier action is a parse-tree in the manner similar to that used in Genetic Programming (Koza, 1992)) as suggested by Wilson (1994b, p.15).

Despite these misgivings, it is nevertheless encouraging that when there is insufficient internal state the ZCS/ZCSM adaptation mechanisms can converge on productive stochastic solutions to the problem of aliased sensory input.

4 Failure on Extended Action-Chains

4.1 Motivation

The results for ZCSM show that internal state can be efficiently exploited to learn profitable sensory-motor mappings in non-Markov environments. However, as was mentioned in the previous section, there is an apparent instability in that the system can occasionally suffer significant failures in performance. Such failures are not evident in the results for ZCS in woods1 or woods7, yet were seen in a number of experiments using ZCSM in environments such as woods101 and woods102. In some cases, the failures were catastrophic: once the system had converged to near-optimal performance, the running average `stpsav` could intermittently return to levels close to those at the start of the trial, before learning had occurred. Different trials in identical conditions (except for alteration of the random number seed) indicated that such failures were unpredictable.

The failures for ZCSM0 in woods101 and woods102 are perhaps unsurprising, given the lack of sufficient internal state. However, it was worrying that similar failures occurred when using ZCSM with sufficient memory for the given environment. Our initial suspicion was that the addition of memory had introduced an unforeseen instability into the adaptation process; hence we were concerned to explore the nature of this instability.

One possibility is that these intermittent failures were caused by *Deceptively Lucky Exploration*. Once the system is (at least partially) adapted to the environment, there is the familiar explore/exploit tradeoff: in some situations a given sense-vector will have several possible action sets. If some of the action sets are relatively strong, they indicate actions which have led to reward in the past, and this past experience can be *exploited* by selecting those actions in order to continue to collect reward. However, it isn't necessarily the case that the weaker actions in the given action set are less productive: they may just have been selected less often, and hence had less opportunity to accumulate reward; thus these weaker action sets should be *explored* occasionally in order to further investigate their effects. The roulette-selection of one action set from those currently available in the match-set is an elegantly simple method of addressing the explore/exploit issue. If the environment is dynamic, then exploration is vital (so as to

continue to adapt in light of changes in the environment). Furthermore, if the environment is non-Markov, then the given sense-vector may be an alias for more than one global state, and the weaker action sets may indicate actions which occasionally lead to a reward. As we saw in Section 3.3, insufficiently large internal state-space could lead to the formation of action sets which correspond to stochastic solutions to the problem of aliased inputs, with the relative distribution of strengths over actions for that state being an indication of the probability of reward following from each action.

We use the phrase *Deceptively Lucky Exploration* to refer to the case where a low-strength action is selected for exploration, and circumstances just happen to be such that the relevant action set receives a high payoff. If this occurs over consecutive trials, the strength of the (previously weak) classifiers in the action set will be increased and the strength of the (previously stronger) classifiers could be reduced, with a corresponding deterioration in overall system performance. That is, a suboptimal move might be converged on if it is selected several times in a row, because the tax on non-selected actions decreases the strengths of better moves. For example, consider two actions a and b for an aliased sense-vector $\vec{v}$ in a non-Markov environment where a is the optimal action in 1% of the global states which generate $\vec{v}$ and b is optimal for the remaining 99% of global states generating $\vec{v}$. Say that the system has adapted so the relative strengths of the actions sets advocating a and b for $\vec{v}$ are 0.01 and 0.99 respectively. Then if by chance there is a *consecutive* sequence of trials in which a is selected and luckily a really is appropriate to the global state, then the relative strength of a could increase to, say, 0.1 while that of b reduces to 0.9; so the system has become more likely to execute action a in situations where really b is appropriate, so performance has degraded. Such degradations of performance would be rare (the reasoning above implies that the severity of degradations would follow a Poisson distribution, so more severe degradations would be less frequent than less severe ones).

Although deceptively lucky exploration may be a problem in environments where it is a possibility, data presented in the next section indicates that ZCS can suffer catastrophic failures even in environments where there is no possibility of exploration interfering with adaptation.

4.2 Experiments with Woods14

Further experimentation with ZCSM in non-Markov environments indicated that the problem became more pronounced in environments with greater optimal expected number of steps-to-food. That is, it seemed that the severity of the failures became more extreme as the average length of the sequential chain of actions leading to reward increased.

In order to test the hypothesis that the problem was due to factors other than deceptively lucky exploration, we designed a Markov environment which is a simple linear path of blank cells through a field of Ts, with an F at one end of the path. We call this environment woods14: see Figure 22.

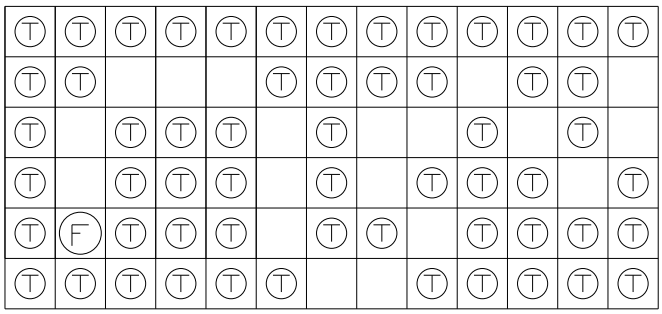

Figure 22: The environment woods14.

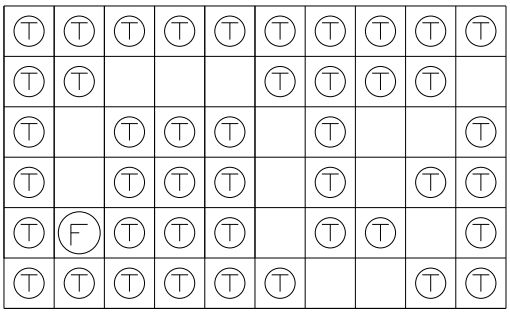

Figure 23: The environment woods14-14.

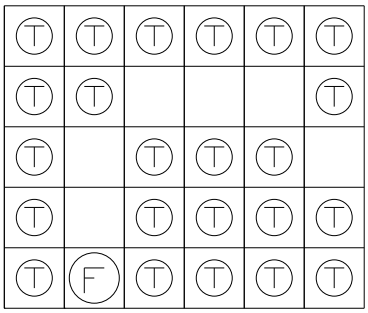

Figure 24: The environment woods14-06.

As can be seen from Figure 22, woods14 has 18 blank cells, and at each cell there is a unique sense-vector, and *only one* action which takes the animat nearer to the F. Because there is only one action in each cell, deceptively lucky exploration cannot occur.

Given its simple nature, we expected that woods14 would represent a fairly trivial learning problem. We planned to first generate some control data using ZCS on woods14, and then to test ZCSM on the same environment to see whether the addition of memory mechanisms had caused the problem: because woods14 is Markov, ZCS should be capable of learning appropriate classifiers, so the comparison between ZCS and ZCSM would be fair.

Our intention in working with woods14 was purely to identify whether chain-length was a major factor underlying the observed instability. For this reason, we did not vary any of the ZCS parameters from the values used in the work reported in Sections 2.2 and 3. To our surprise, we found that even ZCS could not learn reliable strategies for woods14. Apparently the chain-lengths required in this environment are sufficiently long that stable sets of classifiers could not be found.

Problems with long chain maintenance have been noted before in the classifier systems literature. Wilson and Goldberg (1989, p.247) state:

> "...research has suggested that besides being slow to reinforce, bucket brigade chains are also quite fragile in the sense that earlier members tend to have less strength, regardless of the number of sequence repetitions. This seems partly due to a combination of two effects: (1) the probability of reaching payoff starting with an earlier classifier is less than starting with a later one, because of stochastic effects at each step; (2) later classifiers tend to have many sequences leading into them, so they are reinforced more often."

It is our belief that, because ZCS is a minimal system, working with ZCS may further illuminate the reasons for failure to maintain long chains in similar classifier systems.

We therefore embarked on a series of experiments to determine the relationship between chain-length and stable adaptation in ZCS. To do this, we generated a set of environments based on woods14: path (i.e. chain) length was decreased by filling blank cells at the end of the path with Ts. The environments were given names using the format woods14-p where p denotes the number of blank cells in the path. Thus the original woods14 (Figure 22) becomes woods14-18. For illustration, Figures 23 and 24 show two other variants of woods–14.

Each trial with the woods14-p environments was run in the same way as the experiments described in earlier sections of this paper: a blank cell was chosen as a random initial starting position, and the number of steps to F was recorded for each trial in order to generate `stpsav`. Given the simplicity of the woods14-p environments, the theoretical optimum performance is simply calculated from $1 + ((p - 1)/2)$. The expected number of steps under a random-walk strategy was determined empirically for each woods14-p.

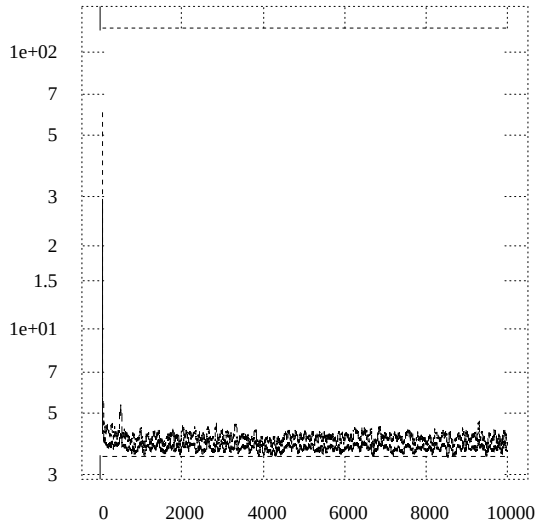

Figure 25: ZCS Scores in woods14-06. Average of 10 trials. Ordinate is trial number. Abscissa: solid line shows mean stpsav; dashed line shows mean plus one standard deviation. Lower horizontal dashed line shows theoretical optimum performance; upper horizontal dashed line shows expected performance using random walk.

Results for using ZCS on the environments woods14-06 to woods14-14 are shown in Figures 25 to 33. As can be seen, while the results for woods14-06 are similar in nature to those for woods1 and woods7, in woods14-09 there are some 'spikes' showing temporary degradation of performance. These become steadily more severe as the path length increases. In woods14-10, there is a spike where average performance is close to that exhibited in the early trials before the system had adapted. For woods14-13 the spikes from the ten trials start to merge into extended periods of degraded average performance, and for woods14-14 the average performance is only very rarely at all close to the theoretical optimum level.

To summarize these data, Figure 34 shows the peak value of stpsav averaged over ten runs, plotted against p for woods14-p. As can be seen, for values of $p > 11$ the worst performance is higher than would be expected from a random search policy. The peak average stpsav values for low p are better than random search, but this is at least in part a consequence of the fact that for low p much of the system's adaptation occurs in the first 50 trials, before stpsav can be calculated.

To some extent there is a loss of clarity caused by averaging the results from several runs. To better illustrate the performance in woods14-13 and woods 14-14, Figures 35 and 36 show stpsav over single runs. As can be seen, on individual runs the system does find sets of classifiers which give near-optimal performance, but there is an instability which can result in such well-adapted classifier sets being disrupted, giving worse-than-random performance. For instance, in Figure 35 there is a sustained sequence of about 500 trials (starting around trial number 8520) where stpsav is worse than random-walk performance.

Given the linear nature of the woods14 environments, there is no opportunity for Deceptively Lucky Exploration to occur. An alternative explanation for the failures is therefore required. By careful analysis of the state of the classifier system during the collapse from near-optimal to worse-than-random performance levels we have been able to determine a likely cause of such failures: collapses in performance in woods14-p environments are due to an interaction between *greedy classifier creation* and *conflicting over-general classifiers*. We discuss each of these in turn below.

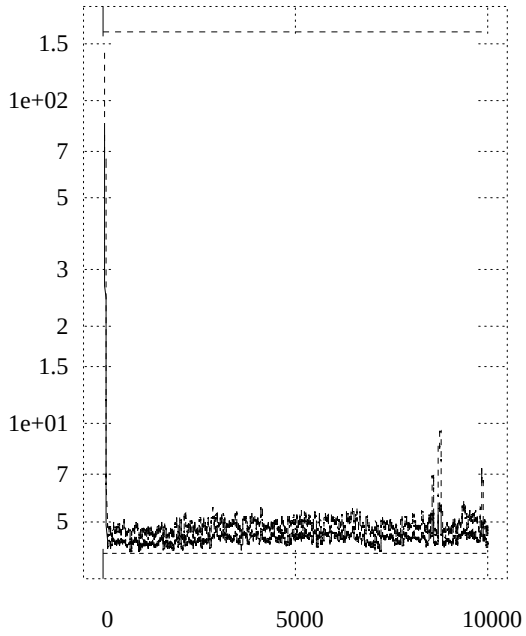

Figure 26: ZCS Scores in woods14-07. Format as for Figure 25.

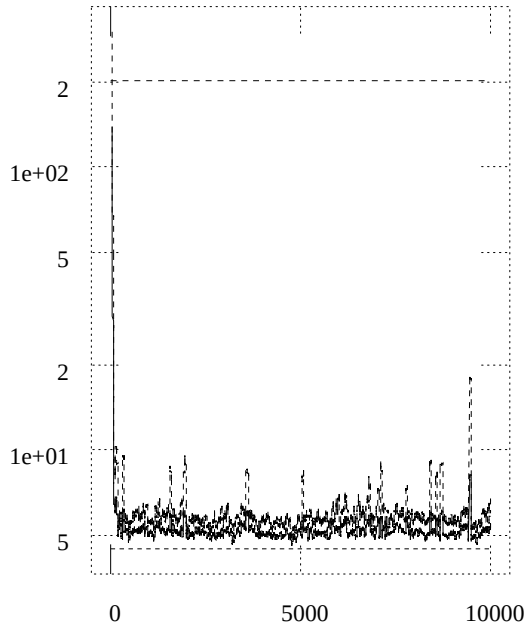

Figure 27: ZCS Scores in woods14-08. Format as for Figure 25.

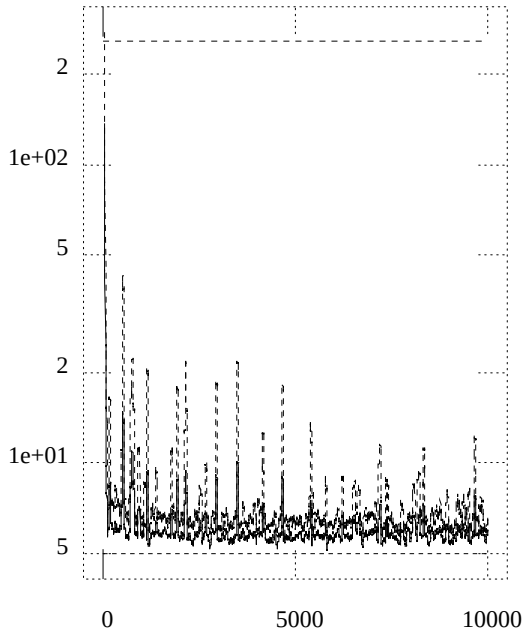

Figure 28: ZCS Scores in woods14-09. Format as for Figure 25.

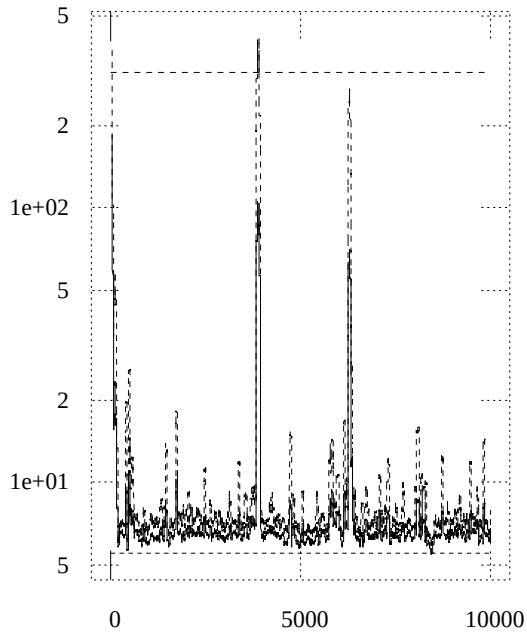

Figure 29: ZCS Scores in woods14-10. Format as for Figure 25.

4.2.1 Greedy Classifier Creation

Although the woods14-p environments may appear superficially unchallenging, the increased incidence of failures as p increases can be understood when one considers the amount of time spent in each cell.

Let us assign identifier labels to each cell: label the cell nearest the F as c_1, and the cell furthest from the F in woods14-p as cell c_p; with the intermediate cells labeled c_i for $i \in \{2, \ldots, p-1\}$. Now on each trial the starting position (i.e. the cell at step zero, $C(0)$) is chosen at random, i.e. $\Pr(C(0) = c_i) = p^{-1}$; $i \in \{1, \ldots, p\}$.

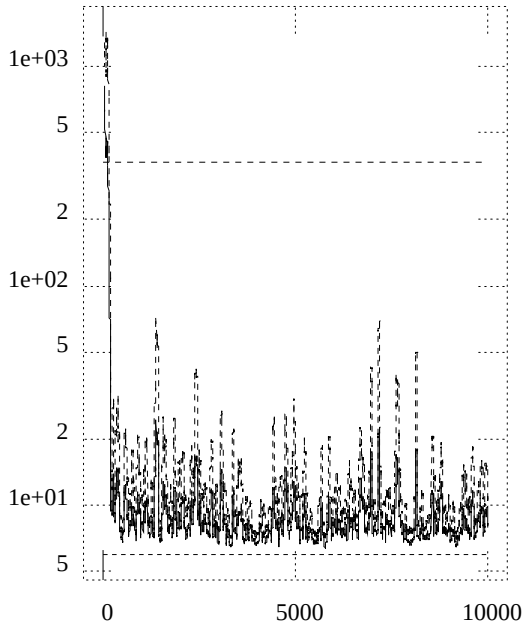

Figure 30: ZCS Scores in woods14-11. Format as for Figure 25.

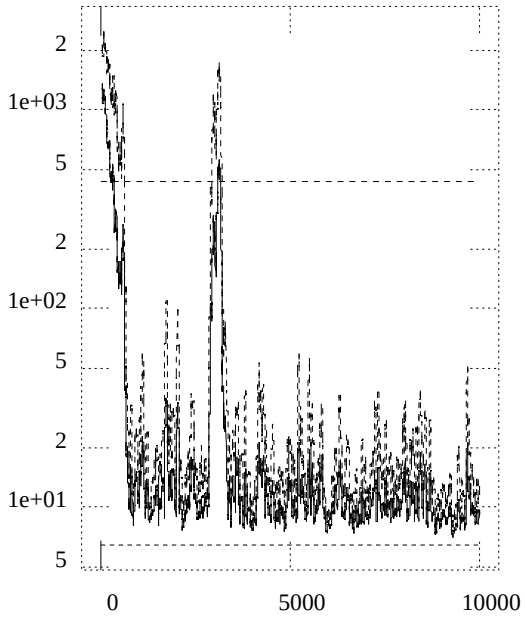

Figure 31: ZCS Scores in woods14-12. Format as for Figure 25.

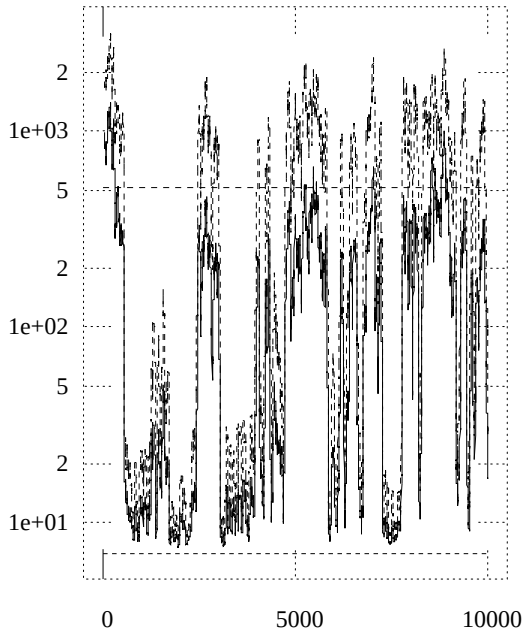

Figure 32: ZCS Scores in woods14-13. Format as for Figure 25.

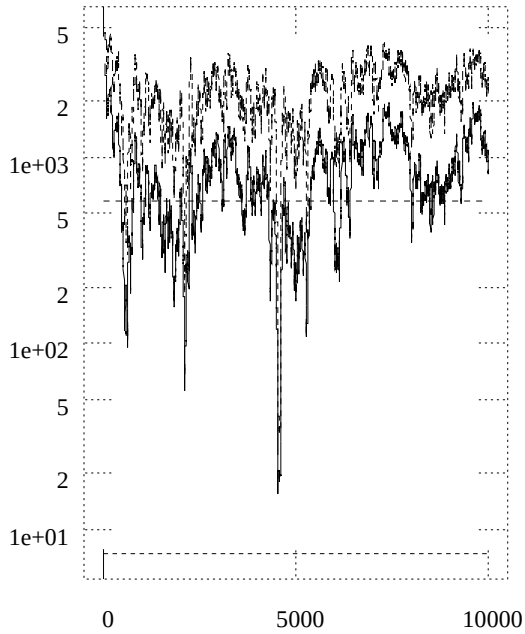

Figure 33: ZCS Scores in woods14-14. Format as for Figure 25.

However, the linear nature of the woods14-p environments *guarantees* that if the animat is at cell c_i then all cells $c_j : j < i$ have to be visited before the trial can end. So when the starting position is chosen uniformly at random, an estimate of the proportion of trials in which cell c_i is visited under an optimal policy is given (as a percentage) by $100(1 + p - i)/p$. Therefore, even under an optimal movement strategy, c_1 is visited on average p times more often than c_p. Furthermore, because c_1 is closest to the reward at F (i.e. at the end of any chain of actions in woods14-p), classifiers matching the sensory input at c_1 will increase in strength most rapidly, and hence be more likely to be selected for reproduction in the GA.

This effect will be exacerbated by the effects of the discount factor γ: classifiers active

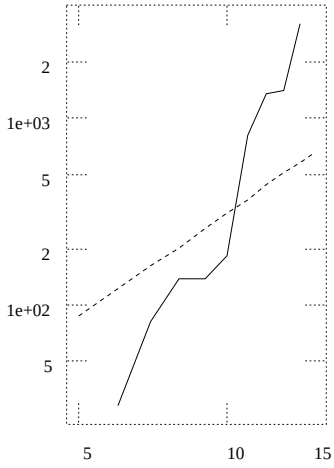

Figure 34: Peak (worst) performance vs. path-length p. Ordinate is path-length. Abscissa is number of steps: solid line is peak `stpsav` value from data presented in Figures 25 to 33; dashed line is empirically derived expected performance of random walk strategy.

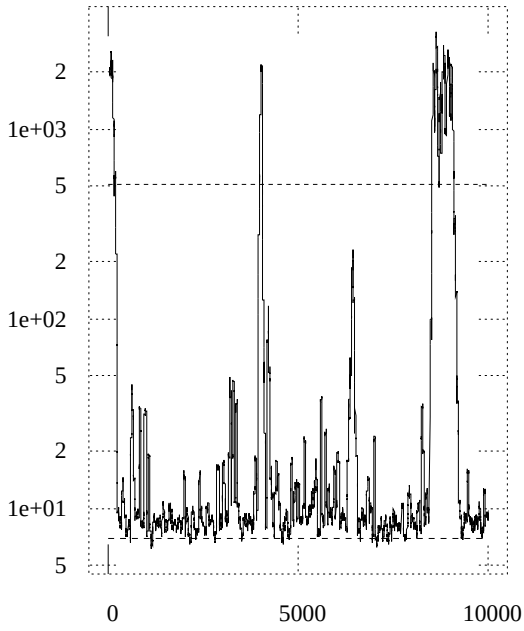

Figure 35: ZCS in woods14-13: `stpsav` over one run.

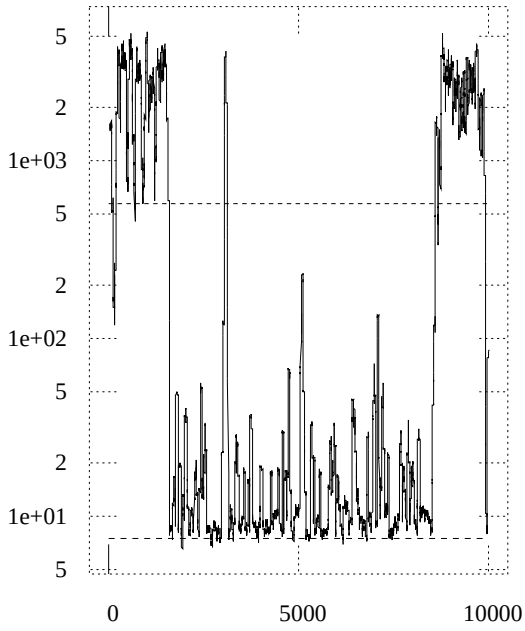

Figure 36: ZCS in woods14-14: `stpsav` over one run.

toward the end of a chain of actions receive more absolute reward than do earlier classifiers in the chain. This is a consequence of the Q-like learning mechanisms in ZCS: Q-Learning is based on optimizing the discounted sum of future rewards $\sum_{j=0}^{\infty} \gamma^j r_{t+j}$ where r_{t+j} is the reward received at timestep $t+j$. This use of γ is a means of giving a higher rating to a reward received recently than to an equal reward received later. Recall that in our experiments $r = 0$ at all timesteps except the final step in a chain, when the animat moves onto a F and $r = r_{\text{imm}}$. If $\gamma \in (0, 1)$, it is fairly easy to show (at least when $[H] = [A]$ on each step in a chain) that the γ discount factors in a chain compound to give exponential decay of absolute reward received with respect to distance from the end of the chain. That is, the reward-based strength increment passed back through the implicit bucket brigade to an action set will be modulated by a factor of γ^{s-1} where s is the number of steps between the action set being active and the reward being

received. In the case of an action set which moves the animat onto a F, $s = 1$ so the reward received is modulated by a factor of 1.0. For an action set active at cell c_p in woods14-p, the reward-based increment will be modulated by a factor of γ^{p-1}: e.g. for $\gamma = 0.71$ and $p = 14$, the compounded discount modulating factor is approximately 0.0117, so action sets in c_1 receive around eighty-five times as much reward as those in c_{14}. The exponential decay may be further exaggerated by the effects of the learning rate β compounding through time in a similar fashion.

So the animat spends proportionately more of its time at cells near the F, and classifiers matching the sensory input at these cells will gain strength and reproduce more frequently than classifiers matching the sensory input for more distant cells, which are earlier in the chains of actions. Moreover, classifiers for the more distant cells will in general be weaker (because rewards passing through the bucket brigade are discounted more heavily as distance from the F increases) and are hence more likely to be selected for deletion. Therefore weak classifiers matching at F-distant cells will be deleted to make way for the offspring of strong classifiers which match at F-near cells. The interaction of these factors leads us to expect that, in a classifier system with a fixed population size, the failures may be caused by such 'greedy classifier creation', i.e. by excessive deletion of classifiers matching at distant cells, to make space for the disproportionate growth of classifiers which match the sensory input close to the F. To demonstrate that this is indeed the case, Figure 37 shows the number of classifiers matching the sense-vector (i.e. the size of the match-set) at each cell $c_i : i \in \{1, \ldots, 14\}$ in woods14-14 with $N = 400$. As can be seen, the cells nearest the F have a disproportionately large number of matching classifiers.

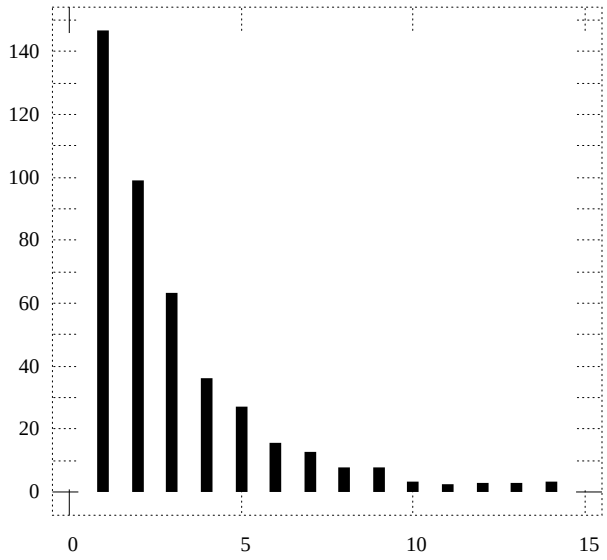

Figure 37: Number of classifiers matching at cell c_i for $i \in \{1, \ldots, 14\}$ in woods14-14 with $N = 400$: ordinate is i; abscissa is average of match-set sizes recorded at trials 2000, 4000, 6000, and 8000 from the run shown in Figure 36.

These data indicate that for woods14-p with large p, a significant factor in the occurrence of catastrophic failures is that cells most distant from the F have comparatively few classifiers matching the sensory input, and these few classifiers are sufficiently weak that they are probable candidates for deletion. If they are deleted, then the system has to re-learn the appropriate action at that cell.

Now consider the most extreme case, where the *only* classifier matching the sensory input at a given F-distant cell is deleted. The next time the animat visits that cell, it will need to generate a new classifier via the covering operation. In the woods14-p environments the probability of randomly choosing the correct action when covering is the reciprocal of the number of possible

external actions: for the experiments described here, this is 0.125 so the probability of randomly choosing an inappropriate action is 0.875. If an inappropriate action is (randomly) chosen for the classifier created in covering, then the system has to repeatedly perform that action until its strength is sufficiently diminished (by repeated decrements during reinforcement) that covering is re-activated. It is not inconceivable that this process (of covering generating a sole inappropriate action which requires many steps to be 'deselected') could be repeated several times in succession within a particular trial, leading to a very large number of steps to F on that trial.

However, once the F has been reached on such a trial in woods14-p, covering *must* have generated the correct action, and so on subsequent trials the problem should be much less likely to occur. Therefore, while greedy classifier generation clearly can account for sudden spontaneous decreases of performance (i.e. one 'bad' trial, taking a large number of steps), it is less clear why the breakdowns in performance are *sustained* over large sequences of trials (because, for the 'bad' trial to have ended, ZCS must have generated classifiers advocating the correct action in *each* cell in the woods14-p environment, so the next trial should not present any problem).

Although it is possible that 'chain reactions' (where inappropriate covering in one distant cell could trigger deletions of weak classifiers at other distant cells, giving rise to further inappropriate covering which causes further deletion of crucial classifiers, and so on) could be the reason for the sustained failures, there is another factor we have identified which appears to play a much stronger role in causing and sustaining the observed failures: this factor is the formation of conflicting over-general classifiers, discussed further in Section 4.2.2. It should be noted that Wilson (1994b, pp.13-15) anticipated the problem of greedy classifier creation in long-chain maintenance, and proposed an extension to ZCS involving a *niche* GA (cf. Booker, 1989) as one potential remedy.

4.2.2 Conflicting Over-General Classifiers

The inclusion of wild-card characters in the classifiers effects a generalization mechanism. Generalization is a powerful feature if there are several global states whose sense-vectors differ in some respects but for which the same action is appropriate: one classifier advocating an appropriate action can match multiple states if it has sufficient wildcards to ignore the differences between the sense-vectors. For example, in the woods environments, if there is a F in a particular cell adjacent to the animat, then a classifier with wildcards at all positions in the sense-vector except for the two bits corresponding to the position of the F can advocate moving in the correct direction (i.e. onto the F, generating a reward) *regardless* of the particular surrounding pattern of Ts and blank cells. Such a classifier should gain a high strength, at the expense of less general classifiers.

However, introducing a generalization mechanism raises the problem of *conflicting over-general classifiers*. A conflicting over-general classifier is one which matches multiple sense vectors for which *different* actions are appropriate. Given the mutual exclusivity of actions in the woods environments, a conflict arises: the action specified by the over-general classifier will be appropriate sometimes, and inappropriate other times. On the occasions when the classifier's action is appropriate it will (eventually) collect reward, and on the occasions when its action is inappropriate it should lose strength. Depending on circumstances, such over-general classifiers may take considerable periods of time for their strengths to diminish sufficiently that they are deleted. Dorigo (1993) refers to such classifiers as *oscillating classifiers*.

Examination of vector fields for woods14-p environments showed that, even when the system had adapted to give near-optimal performance, the more distant cells often had match-sets which strongly advocated wholly inappropriate actions, i.e. moving away from the F or into a neighboring T. Figure 38 shows one such vector field for woods14-13 (at trial number 8543, same run as illustrated in Figure 35). As can be seen, for $i > 7$, cells c_i have match-sets which

advocate inappropriate actions.

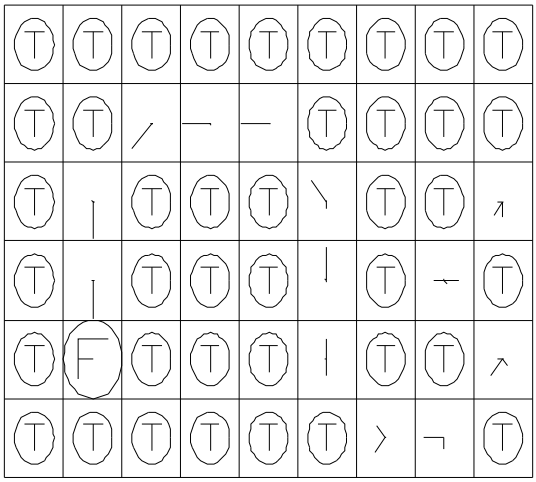

Figure 38: Over-generalization in woods14: vector field generated in woods14-13 at trial number 8543 in the run shown in Figure 35. See text for discussion.

Inspecting the classifiers advocating these inappropriate actions, two things were clear: their strength was significantly higher than the strength expected if they had been generated by covering, and they were sufficiently general to match the sense-vectors at cells *nearer* the food. On reflection, this is not a particularly unlikely occurrence. In the woods14-p environments, the sense-vectors for all cells $c_i : 1 < i < p - 1$ will encode six surrounding T cells and two blank cells, with two bits encoding the contents of each cell. Such regularities imply that the sense-vectors for different cells may only vary in the contents of one or two cells. For example: cells c_2 and c_7 are only distinguishable by the contents of the north and north-west cells (see Figure 39); therefore the Hamming distance between their binary-encoded sense-vectors is 4, and so it is possible to create an over-general classifier with four wildcards which will match the sensory input at both cells, yet the appropriate actions in the two cells are *opposites*: in the terminology of Yates and Fairley (1993), the over-general classifiers *subsume* the more specific classifiers active at F-distant cells.

The problem of over-general classifiers is significantly worsened by greedy classifier creation: classifiers active at cells near the F will rapidly gain strength and be selected for breeding, where more general classifiers can be created. As sense-vectors of the F-near cells are only a short Hamming distance from the sense-vectors of more distant cells, strong over-general classifiers can be created which advocate incorrect actions in distant cells, which (being earlier in the chain of actions) generally have weaker classifiers. Hence the over-general classifiers which generate

```
T  T  .        T  #  #        T  .  T
T  *  T        T  *  T        T  *  T
T  .  T        T  .  T        T  .  T
```

Figure 39: Over-generalization in woods14-$p : p > 8$. Left-hand figure: surrounding cells when animat (*) is at cell c_2: appropriate action is to move south. Right-hand figure: surrounding cells when animat is at cell c_7: appropriate action is to move north. Center figure: an over-general classifier which matches at both c_2 and c_7: "#" denotes cells whose corresponding pair of bits in the classifier condition are both wildcards.

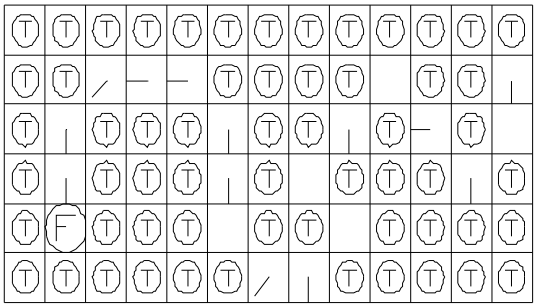

Figure 40: Over-generalization in woods14: vector field generated in woods14-18 from a population of classifiers exposed only to woods14-04 for 5000 trials. Cells c_5 to c_{18} should be blank, because the system has not been exposed to their sense-vectors. Over-general classifiers generate inappropriate actions in several of the previously "unseen" cells.

Cell in woods14-04	Cell in woods14-18	Sensory Hamming Distance
c_1	c_7	1
	c_6	5
c_2	c_6	4
	c_7	4
	c_{10}	4
	c_{13}	4
	c_{16}	4
c_3	c_9	4
c_4	c_5	2
	c_{15}	6

Figure 41: Table showing cells in woods 14-18 affected by adaptation to woods 14-04, and the Hamming distance between the sense vectors for those cells. Note that c_4 in woods14-04 is the most distant cell and hence has a T in the position occupied by c_5 in woods14-$p : p > 4$. Hence the Hamming distances from c_5 and c_{15} in woods14-18 to c_4 in woods14-04 are less than the respective distances measured in woods14-$p : p > 4$.

appropriate actions nearer the F are inappropriately favored by the action selection process at distant cells, and their high strengths mean that it takes proportionately more steps before their strengths are sufficiently reduced to allow other, more appropriate, actions to be generated.

To demonstrate the effects of over-general classifiers, Figure 40 shows a vector field for woods14-18 generated from a population of classifiers which had been trained for 5000 trials on woods14-04. As would be expected, cells c_1 to c_4 have appropriate classifiers. However, the previously 'unseen' cells $c_i : i > 4$ have inappropriate actions strongly advocated in the match-sets. Examination of the classifiers active in the previously 'unseen' cells revealed that they were over-general classifiers which also matched one or more of the 'seen' cells $c_i : i \leq 4$. In this particular experiment, cells $c_i : i \in \{5, 6, 7, 9, 10, 13, 15, 16\}$, all have inappropriate actions advocated by classifiers formed in the adaptation to woods14-04. The table in Figure 41 illustrates which cells in woods14-04 have affected cells in woods14-18 over a number of such trials, and indicates the Hamming distance between the sense vectors.

It is also worth noting in Figure 40 that c_{18} has also been affected by the exposure to woods14-04, but the effect has been that an *appropriate* action is advocated: here an over-general classifier

has a beneficial side-effect.

4.2.3 Summary

The combined effects of greedy classifier creation and over-general classifiers are the primary causes of the collapses in performance that we have analyzed on a step-by-step basis. The worse-than-random performance times for the system when initially adapting and when re-adapting after a collapse seem to be due mainly to the large number of steps required to diminish the strength of inappropriate over-general classifiers to such a degree that covering can take effect in the F-distant cells. Similar trials with ZCSM1 and ZCSM2 in woods14-p environments indicated that the addition of memory offered no improvement: in fact, the results were significantly worse.

Although the fragility of long chains has previously been discussed in the classifier systems literature (e.g. Wilson & Goldberg, 1989; Compiani, Montarini, & Serra, 1990), we believe that the experiments reported here are informative because they demonstrate that the fragility is present even in the minimalist ZCS. Furthermore, the effects of conflicting over-general classifiers may reveal a more fundamental issue: the over-general classifiers are retained in the population because there is no provision in ZCS for penalizing classifiers lacking accuracy or consistency. As Wilson (1994a) notes, other authors have discussed the need for including such mechanisms (see e.g. Holland, 1976; Holland & Reitman, 1978; Frey & Slate, 1991; Grefenstette, Ramsey, & Schultz, 1990; Dorigo, 1993). Most recently, Wilson (1994a) describes a classifier system, XCS, which is based on ZCS with extensions to allow classifier fitness to be based on accuracy. Our results indicate that, at least in applications requiring long chains of actions, XCS should be more efficient and robust than ZCS.

5 ZCS in a continuous-space environment with distal sensing

The results of the previous sections appear to indicate that the prospects of using ZCS and/or ZCSM in animat guidance are limited by the problems of long chain maintenance. A final possibility we explored was whether these problems are due to the cellular nature of the simple 'grid-world' woods environments. Such environments have been criticized in the past for, amongst other things, their tenuous links to the real physical environments faced by animals and robots, their treatment of time and space as discrete variables, and their conflation of sensing and recognition (Brooks, 1992; Cliff & Bullock, 1993). Nevertheless, it is also sometimes said (e.g. Brooks, 1992, p.5) that the lack of realistic noise and stochastic processes makes the dynamics of cellular worlds *more* brittle than those of the real world. For these reasons, it is tempting to test ZCS in more challenging conditions than those offered by the woods environments. In this section we briefly present results which indicate that the problems of ZCS are not necessarily eased in more complex environments.

In our more complex environments space is a continuous variable. The animat is still considered as a 2-dimensional (2-D) entity traversing a plane, but movement is uncertain. The animat is a circular agent whose circumference is divided equally into a number of radially oriented 'distal sensors' which can be used to detect the presence of circular obstacles scattered around the planar space. The distal sensors are idealized and grossly simplified one-bit binary 'photodetector' units, all with the same fixed threshold. If more than this threshold proportion of their 'angle of acceptance' is occupied by obstacles then their output is 0, otherwise 1. This 'flatland vision' is effected by solving the relevant 2-D projection equations:[6] see Figure 42. Although simplistic, such a system does provide a facility for examining the effects of limited visual acuity, occlusion and disocclusion, etc. For brevity, we refer to these studies as FLAVA experiments (from FLAtland Vision for Animats). One of the primary inspirations for our FLAVA experiments was

[6] Further details of FLAVA, including copies of the C code, are given in (Cliff, 1994).

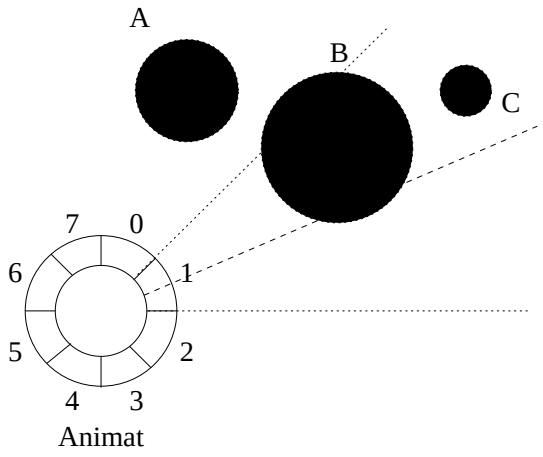

Figure 42: Schematic illustration of the flatland vision system. The circular animat is shown with 8 equal-angle 'visual' sensors occupying its circumference. The dark circles are obstacles in the world. The dashed line indicates the 'optical axis' of sensor 1, and the dotted lines indicate the limits of its angle of acceptance. Obstacle A has no effect on sensor 1 because it is outside the angle of acceptance.. Obstacle B subtends over half the acceptance angle of sensor 1. Obstacle C has no effect on sensor 1 because it is occluded by obstacle B: if B was removed, C may be too distant to subtend a sufficient proportion of the sensor's angle of acceptance to affect it.

prior research in modeling visual localization and navigation in bees and wasps (Cartwright & Collett, 1983, 1987; Collett, 1992), which used similarly minimal models.

In the current FLAVA work, time proceeds in discrete timesteps. On each timestep, the animat makes a movement: the movement is a step of fixed length, whose direction is chosen randomly within some range specified by the classifier's external action. For compatibility with our earlier experiments, we gave the ZCS-FLAVA animat 8 external actions: as with the cellular woods environments, the action specifies one of eight directions of movement, with 45° between successive directions. On each step, a uniform random deviate over [-22.5°,22.5°] is added to the specified movement direction before the step is taken. The net effect is that an animat applying a constant external action will wander in a consistent direction with random drift: a biased "drunkard's walk" over a continuous space.

The animat can move in the two spatial dimensions, but maintains a constant orientation. It cannot move into space occupied by an obstacle. It has a one-bit omnidirectional "collision detector" which outputs 1 if the animat attempts to collide with an obstacle, and 0 otherwise. There is no noise or uncertainty in either the 'visual' sensors or the 'collision detector'. Drawing inspiration from studies of visual navigation in animals, the location of rewards in our FLAVA studies are *not* detectable by the animat's sensors: the animat has to find its way to a reward by reference to 'visual landmarks' alone.

Further realism is introduced by making the length of each random movement-step small in comparison to the size of the world: sequences of small movement steps can be considered as an approximation to spatiotemporally continuous movement. However, this approach raises the problem of the temporally discrete nature of the classifier system: ZCS, like most other classifier systems, operates in discrete timesteps, each composed of one run through the sense-act-reinforce-discovery sequence. If Δt represents the inter-timestep interval, then reducing Δt (to get a better approximation to continuous movement) compresses the timescale over which chains of actions can be supported. To avoid this problem, we use ZCS in a *sensory interrupt* fashion: once an external action is selected, that (noisy) action is applied on each timestep in a pure sense-act loop (i.e. *without* reinforcement or discovery) which is only interrupted when

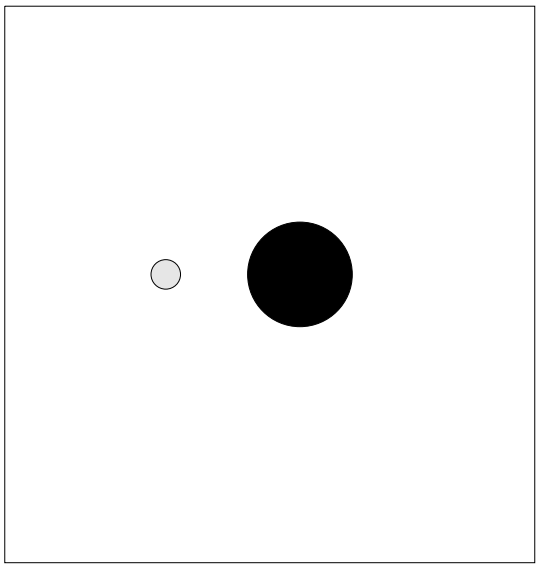

Figure 43: A simple FLAVA environment. The large dark circle is an obstacle; the small light circle denotes the zone where the animat will receive reward (the animat cannot sense this zone). See text for discussion.

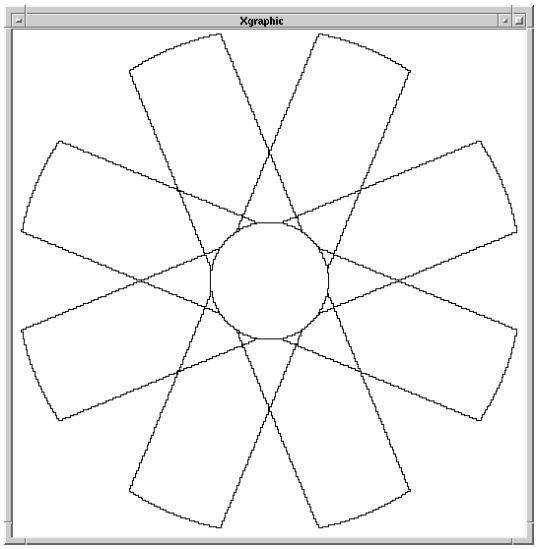

Figure 44: Sensory signature neighborhoods for the environment illustrated in Figure 43. See text for discussion.

either a reward is received (in which case the trial ends) or when the sense-vector alters (either by a change in the 'visual' inputs or by the collision detector becoming active). When a sensory interrupt occurs, reinforcement and discovery may take place, followed by roulette-selection of a new action; the system then re-enters the pure sense-act cycle. This interrupt technique employs ZCS as a system which selects a new action only when the sense-vector changes, rather than at the end of each movement step. Classifier actions now specify changes in movement policy, rather than actual changes in position, and hence short chains of actions may underlie temporally extended sequences of movement steps, consisting of arbitrary numbers of Δt intervals.

Figure 43 shows a very simple FLAVA environment. The world measures 52×52 (arbitrary) distance units, and the animat takes steps 0.5 units long. There is a single small patch of reward (radius=1.0) 12 units west of a single circular obstacle (radius=5.0) which might act as a 'landmark' for locating the reward. The animat has 8 'visual' sensors, all with threshold 0.5, arranged as illustrated in Figure 42, and has a radius of 1.0 units.

Figure 44 shows a map of this environment's sensory signature neighborhoods (cf. Mahadevan and Connell (1992, p.334)): the lines partition the 2-D environment into neighborhoods; the sense-vector is identical at all points within each neighborhood (recall that the reward cannot be detected by the animat's sensors). Because there is only one circular obstacle, and the animat's visual sensors are radially symmetric, the signature map is also radially symmetric. At the outer edges of the environment, the obstacle does not subtend a sufficient proportion of the acceptance angles of any photoreceptor units to register. As the animat moves closer to the obstacle, there are large neighborhoods where one of the eight photoreceptors registers the obstacle, smaller neighborhoods where two photoreceptors register, and some very small neighborhoods where three register. The sensory signature neighborhoods can be considered as defining a particular non-Cartesian 'grid-world' around which the animat moves. Within any signature neighborhood, the sense-vector is identical and hence the animat *cannot* determine its precise location in the world; the best it can do is determine an appropriate (adaptive) action. Readers of a constructivist philosophical persuasion will have noticed that the grid is defined by

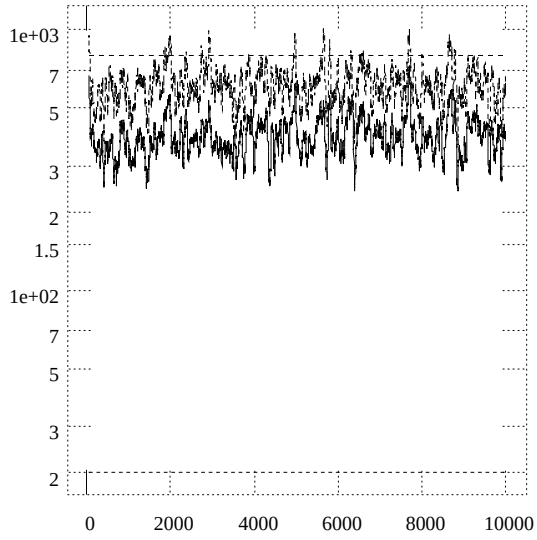

Figure 45: ZCS scores in the FLAVA environment of Figure 43. Solid line is mean (over ten runs) of a running average (over 50 trials) of trial path distance; dashed line is mean plus one standard deviation. Upper horizontal dashed line is average behavior of random action choice. Lower horizontal dashed line is lower bound on optimal behavior. See text for discussion.

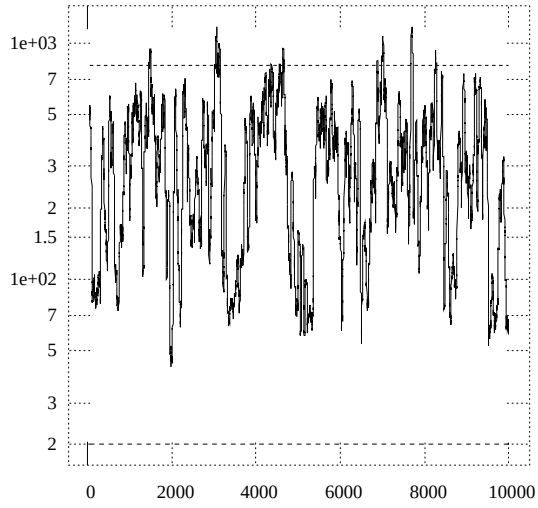

Figure 46: ZCS performance in a single trial. Solid line is running average of path distance over last 50 trials. See text for discussion.

the *interaction* between the animat's sensory sampling strategy (cf. Cliff & Bullock, 1993) and its environment: there is no 'objective reality' for the agent that is independent of the agent; if it had a different number of sensors, or a different arrangement of 8 sensors, then the signature neighborhoods, and hence the space of possible agent-environment interactions, could be very different.

Because the obstacle is 'invisible' from all the edges of the world, toroidal wraparound can still be used if the animat tries to move off an edge of the environment: the effect of the wraparound is to tile the 2-D plane with copies of this environment, extending infinitely in all directions.

As with the woods experiments, each trial starts with animat at a random location (neither colliding with an obstacle or within the reward zone). Each trial ends when the animat moves onto the reward zone. The measure of performance on a trial is the line integral of the animat's trajectory, i.e. total distance traveled from the start location to the point of contact with the reward zone. The average distance using random selection of action when a sensory interrupt occurs is approximately 790 units. The average distance under an optimal movement strategy with perfect information in the absence of obstacles is approximately 20 distance units: this serves as a lower bound on the average optimal performance in the presence of obstacles.

Figure 45 shows results from ZCS adapting to this environment. The performance measure is a running average of path distance over the last 50 trials. As can be seen, there is little evidence of adaptation having occurred, but the scores are better than random performance. Once again, this is due to the system converging on well-adapted populations of classifiers which then undergo catastrophic failures. Figure 46 shows the running average of path distance over one trial. As can be seen, the transients between adapted and non-adapted behaviors are very brief.

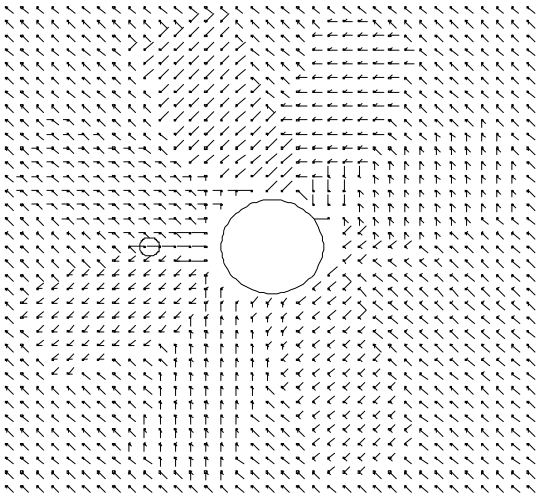

Figure 47: Vector field for ZCS in the FLAVA environment of Figure 43, from the run illustrated in Figure 46, after trial 5340. Vectors generated on a regular grid: dot indicates vector basepoint.

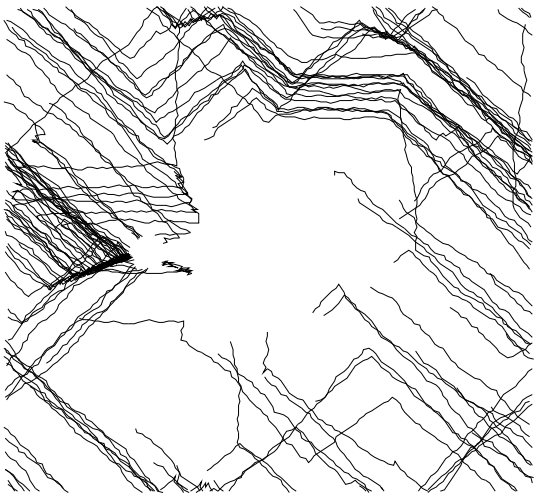

Figure 48: Fifty individual trial trajectories, trials 5290 to 5340 of the run illustrated in Figure 46. The dense concentration of trajectories leading to the reward zone from the south-west is caused by the animat alternating between two adjacent signature neighborhoods.

The behavior produced by populations of well co-adapted classifiers can be visualized using vector fields to show average expected behavior, and plots of individual trial trajectories to show actual generated behavior. Results from deriving a vector field at regularly spaced points over the environment are shown in Figure 47: as can be seen, the signature neighborhoods near the reward zone have appropriate vectors, but some of the neighborhoods more distant from the reward have vector distributions that are not distinguishable from the 'background' distribution in the perimeter neighborhood, where the obstacle is not 'visible'.

While this is not necessarily a sign of poor adaptation, examining individual trial trajectories, as illustrated in Figure 48, indicates a problem. ZCS has developed sets of classifiers for sense-vectors near to the reward zone which advocate actions that 'bounce' the animat between adjacent signature neighborhoods (i.e. the action in a neighborhood $\mathcal{N}_1$ sends the animat to some other neighborhood $\mathcal{N}_2$, and the action in $\mathcal{N}_2$ sends the animat back to $\mathcal{N}_1$). While this clearly can lead the animat to the reward, each transition between neighborhoods generates a sensory interrupt and hence counts as a step in a chain of actions, so the problem of long-chain maintenance re-occurs; classifiers active in signature neighborhoods more distant from the reward zone will be deleted to make way for classifiers active in neighborhoods nearer the reward.

That this problem occurs in the simple environment of Figure 43 is worrying: as more obstacles are added, the number of signature neighborhoods is likely to grow dramatically: Figure 49 shows an environment with just three obstacles. As can be seen, the number of signature neighborhoods has increased markedly from the 25 neighborhoods of the single-obstacle world shown in Figure 43. As the number of neighborhoods grows, so does the need for dealing with long chains of actions, because even when ZCS is driven by sensory interrupts each transition across a neighborhood counts as one action. If we allow noise in the sensors, then sensor values may fluctuate *within* signature neighborhoods, which may require that the sensory interrupt approach is abandoned. But if the interrupt approach is abandoned then the problem of the Δt tradeoff between approximate spatiotemporal continuity of motion and the temporal compression of action chains requires some alternative solution. From these observations, we conclude that the

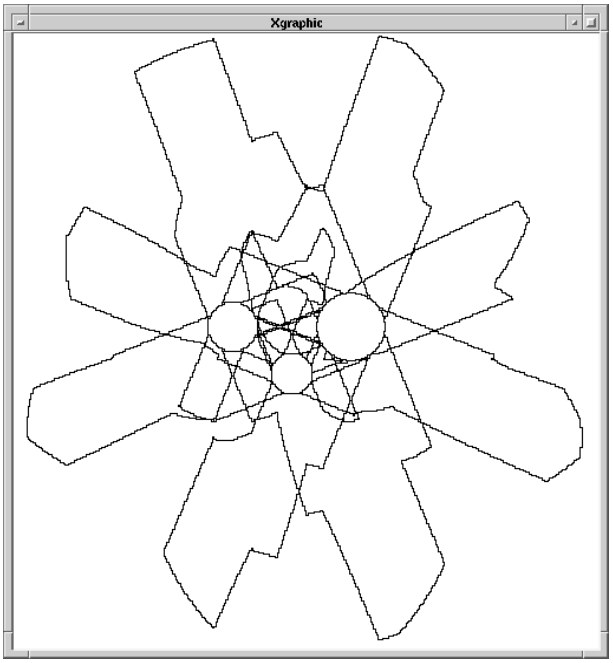

Figure 49: Sensory signature neighborhoods for a FLAVA world with three circular obstacles. See text for discussion.

problems faced by ZCS are not artifacts of the cellular woods environments.

6 Conclusions

Our overall conclusion is that, while ZCS plays an important role in bringing out the similarities between classifier system learning and Q-Learning, its minimalism may cause problems if it is to be used in applications requiring either large amounts of internal state, or the maintenance of long chains of actions. However, these two issues are problematic for most classifier systems, so ZCS is no worse than many other systems; moreover, its minimalism offers clarity in determining the causes of the problems, and in assessing the effects of proposed remedial mechanisms.

To summarize, there are three significant contributions to the literature in this paper.

First, we have demonstrated a reconstruction of ZCS as described by Wilson (1994b) and have shown that Wilson's published results can be replicated. Replication of published results is an important (if unglamorous) task in any scientific field.

Second, we have implemented Wilson's proposal for adding temporary memory to ZCS. Although our results demonstrate that Wilson's proposals work when the number of bits of temporary memory is small, combinatorics indicate that the approach will not scale well as the number of bits required increases. Nevertheless, our demonstration that ZCS could converge on stochastic solutions when there was insufficient internal state to disambiguate sensory aliases implies that there may be applications for which insufficient memory is not an insurmountable problem.

Third, we have demonstrated the limits of performance for ZCS in maintaining long chains of actions. It is informative that even the minimalist ZCS suffers from fragility reported in other, more complex classifier systems. The causes of the fragility lead us to conclude that ZCS in its basic form has an urgent need for the incorporation of mechanisms which will alleviate the problem: without such remedial mechanisms, the prospects for using ZCS for anything more advanced than generating primitive adaptive behaviors in simple cellular worlds would appear

to be poor.[7] One plausible mechanism is to allow accuracy to play a role in calculating classifier fitness: Wilson's recent work on XCS (Wilson, 1994a) is clearly a promising development in this direction.

Acknowledgements

The authors thank Stewart Wilson for his interest and encouragement over the course of this work, and especially for his comments on earlier drafts of this paper. Thanks also to Jean-Arcady Meyer for his comments on an earlier draft. Thanks to Malcolm McIlhagga and Hilary Tunley for allowing use of their SPARC machines. Financial support for Susi Ross came from a UK SERC/EPSRC MSc studentship grant.

Reference

Booker, L. B. (1989). Triggered rule discovery in classifier systems. In Schaffer, J. D. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 265–274. Morgan Kaufmann.

Brooks, R. A. (1992). Artificial life and real robots. In Varela, F. J., & Bourgine, P. (Eds.), *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life (ECAL91)*, pp. 3–10 Cambridge MA. M.I.T. Press Bradford Books.

Cartwright, B. A., & Collett, T. S. (1983). Landmark learning in bees: Experiments and models. *Journal of Comparative Physiology*, *151*, 521–543.

Cartwright, B. A., & Collett, T. S. (1987). Landmark maps for honeybees. *Biological Cybernetics*, *57*, 85–93.

Cliff, D. (1994). Flava: Flatland vision for animats; a simple minimal simulation of distal sensing for animat studies. Csrp 339, University of Sussex School of Cognitive and Computing Sciences.

Cliff, D., & Bullock, S. G. (1993). Adding 'foveal vision' to Wilson's animat. *Adaptive Behavior*, *2*(1), 47–70. Also available as University of Sussex School of Cognitive and Computing Sciences Technical Report CSRP263.

Collett, T. S. (1992). Landmark learning and guidance in insects. *Proc. R. Soc. Lond. B*, *337*, 295–303.

Colombetti, M., & Dorigo, M. (1994). Training agents to perform sequential behavior. *Adaptive Behavior*, *2*(3), 247–276.

Compiani, M., Montarini, D., & Serra, R. (1990). Learning and bucket-brigade dynamics in classifier systems. *Physica D*, *42*, 202–212.

Dorigo, M. (1993). Genetic and non-genetic operators in ALECSYS. *Evolutionary Computation*, *1*(2), 151–164.

Dorigo, M., & Bersini, H. (1994). A comparison of Q-learning and classifier systems. In Cliff, D., Husbands, P., Meyer, J.-A., & Wilson, S. W. (Eds.), *From Animals to Animats 3: Proceedings of the third international conference on simulation of adaptive behavior*, pp. 248–245. MIT Press Bradford Books.

[7] Such problems are not unique to ZCS: similar issues arise when most reinforcement learning techniques are applied to situated domains: see (e.g. Matarić, 1994).

Frey, P. W., & Slate, D. J. (1991). Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, *6*, 161–182.

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

Grefenstette, J. J., Ramsey, C. L., & Schultz, A. C. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning*, *5*(4), 355–382.

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

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

Holland, J. H., & Reitman, J. S. (1978). Cognitive systems based on adaptive algorithms. In Waterman, D. A., & Hayes-Roth, D. (Eds.), *Pattern-Directed Inference Systems.* Academic Press, New York.

Koza, J. (1992). *Genetic Programming: On the programming of computers by means of natural selection.* MIT Press.

Littman, M. L. (1994). Memoryless policies: Theoretical limitations and practical results. In Cliff, D., Husbands, P., Meyer, J.-A., & Wilson, S. W. (Eds.), *From Animals to Animats 3: Proceedings of the third international conference on simulation of adaptive behavior*, pp. 238–245. MIT Press Bradford Books.

Mahadevan, S., & Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, *55*, 311–365.

Matarić, M. J. (1994). Reward functions for accelerated learning. In Cohen, W., & Hirsh, H. (Eds.), *Machine Learning: Proceedings of the Eleventh International Conference* San Francisco, CA. Morgan Kauffman.

Ross, S. (1994). *Accurate Reaction or Reflective Action? Experiments in adding memory to Wilson's ZCS.* Master's thesis, University of Sussex School of Cognitive and Computing Sciences. Unpublished.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards.* Ph.D. thesis, University of Cambridge. Unpublished.

Watkins, C. J. C. H., & Dyan, P. (1992). Technical note: Q-learning. *Machine Learning*, *8*, 279–292.

Whitehead, S. D., & Ballard, D. H. (1990). Learning to perceive and act. Technical report TR 331, University of Rochester Computer Science Department, Rochester, NY. (Revised Edition).

Whitehead, S. D., & Lin, L.-J. (1993). Reinforcement learning of non-Markov desision processes. *Artificial Intelligence*, .(.). To appear 1995.

Wilson, S. W. (1985). Knowledge growth in an artificial animal. In Grefenstette, J. J. (Ed.), *Proceedings of an International Conference on Genetic Algorithms and their Applications (ICGA85)*, pp. 16–23 Hillsdale: New Jersey. Lawrence Erlbaum Associates.

Wilson, S. W. (1987). Classifier systems and the animat problem. *Machine Learning*, *2*, 199–228.

Wilson, S. W. (1991). The animat path to AI. In Meyer, J.-A., & Wilson, S. W. (Eds.), *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior (SAB90)*, pp. 15–21. M.I.T. Press Bradford Books, Cambridge MA.

Wilson, S. W. (1994a). Classifier fitness based on accuracy.. Submitted to *Evolutionary Computation*.

Wilson, S. W. (1994b). ZCS: A zeroth level classifier system. *Evolutionary Computation*, *2*(1), 1–18.

Wilson, S. W., & Goldberg, D. E. (1989). A critical review of classifier systems. In Schaffer, J. D. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 244–255. Morgan Kaufmann.

Yates, D. F., & Fairley, A. (1993). An investigation into possible causes of, and solutions to, rule strength distortion due to the bucket brigade algorithm. In Forrest, S. (Ed.), *Proceedings of the fifth international conference on Genetic Algorithms*, pp. 246–253. Morgan Kaufmann.