Introducing a Genetic Generalization Pressure to the Anticipatory Classifier System - Part 2: Performance Analysis

Martin V. Butz
Dep. of General Engineering
University of Illinois
Urbana-Champaign, IL 61801
butz@illigal.ge.uiuc.edu

David E. Goldberg
Dep. of General Engineering
University of Illinois
Urbana-Champaign, IL 61801
deg@illigal.ge.uiuc.edu

Wolfgang Stolzmann
Institute for Psychology III
University of Wuerzburg
Germany
stolzmann@psychologie.uni-wuerzburg.de

Abstract

The Anticipatory Classifier System (ACS) is able to form a complete internal representation of an environment. Unlike most other classifier system and reinforcement learning approaches, it is able to learn latently (i.e. to learn in an environment without getting any reward) and to form an internal model of the perceived environment. After the observation that the model is not necessarily maximally general a genetic generalization pressure was introduced to the ACS. This paper focuses on the different mechanisms in the anticipatory learning process, which resembles the specialization pressure, and in the genetic algorithm, which realizes the genetic generalization pressure. The capability of generating maximally general rules and evolving a completely converged population is investigated in detail. Furthermore, the paper approaches a first comparison with the XCS classifier system in different mazes and the multiplexer problem.

1 INTRODUCTION

The ACS is not the first approach which forms a complete internal model (i.e. a cognitive map) of the external environment. Reinforcement learning approaches, like Dyna (Sutton, 1991), form an internal model but are not able to generalize and indeed form an identical internal representation of the outside world. Riolo (1991) showed that classifier systems are also able to form a cognitive map and thus are able to learn latently and do lookahead planning. However, the representation of the resulting state in Riolo's CFSC2 is hidden in the rules and message list of the original learning classifier list structure (Booker, Goldberg, & Holland, 1989). Moreover, there was no generalization process in CFSC2, either. Drescher (1991) published a Context/Action/Result-unit approach, he called schemata. His approach is based on the Piagetian development theory. One of his foundational problems is how to notice the result of an action. This problem is solved in the ACS by using the mechanism of anticipatory behavioral control proposed by Hoffmann (1993). The similar mechanism in the ACS is called the Anticipatory Learning Process (ALP). Rules are formed by first considering the perceptual consequences of an executed action and after that differentiating the situations in which these consequences occur.

With the ALP, a specialization process is present in the ACS. The ACS starts with a completely general knowledge and incrementally specializes it as far as necessary. Butz, Goldberg, and Stolzmann (1999) first discussed the problem that the ACS is not necessarily generating an accurate, maximally general knowledge but it is sometimes over-specializing. This pressure towards over-specialization was now overcome by using a genetic algorithm (GA). The resulting genetic generalization pressure (Butz, Goldberg, & Stolzmann, 2000) enables the ACS to converge to the accurate, maximally general rules. However, the convergence was only illustrated in one simple task and the different mechanisms and constants were not investigated in detail which is the aim of this paper.

First of all we will summarize the different procedures and involved parameters in the ACS. Section 3 will investigate the ALP and its enhancement to the whole action set. After that, Section 4 discusses the usefulness of mutation and crossover. Section 4.3 will illustrate the amount of convergence on a rather untypical problem for the ACS, the multiplexer problem. The application in this environment will show that the ACS is able to realize a similar convergence ratio as the XCS classifier system. After that, Section 4.4 compares the reinforcement learning abilities of the ACS to the one in XCS. Finally, a conclusion is given.

2 THE ACS WITH GA

The actual ACS still has its basic structure, which was introduced by Stolzmann (1997). Stolzmann (1999) published enhancements in the ALP. Finally, Butz, Goldberg, and Stolzmann (2000) introduced the enhancement of the application of the ALP and a genetic algorithm to the ACS. Here we revisit the current structure, the integrated processes and the involved constants.

2.1 THE BASIC STRUCTURE

Basically, an ACS always interacts with an environment. At each time step t it perceives a state $\sigma(t) \in \{s_1, s_2, ..., s_m\}^L$, executes an action $\alpha(t) \in \{r_1, r_2, ..., r_n\}$ and receives a reward $\rho(t) \in IR$. m are the number of different possible detector values, s_i the different detector values, L the number of detectors, n the number of different executable actions, and r_i the different actions.

The ACS presents its knowledge in rules, which are called classifiers. A classifier consists out of a condition part C, an action part A, an effect part E and a mark M. $C, E \in \{s_1, ..., s_m, \#\}$, $A \in \{r_1, ..., r_n\}$, and $M = (m_1, ..., m_L)$ with $m_i \subseteq \{s_1, ..., s_m\}$. A '#'-symbol in C (i.e. a 'don't-care' symbol) means that the ACS ignores the corresponding detector information while a '#'-symbol in E (i.e. a 'pass-through' symbol) means that the ACS believes that the corresponding detector value remains the same after the execution of the action r. Furthermore, each classifier has both a quality $q \in [0,1]$ (i.e. the measure for the accuracy of the anticipations of a rule) and a reward measure $r \in IR$ (i.e. the measure for the predicted payoff).

A behavioral act forms at first a match set M_{set} out of the current population considering the current state $\sigma(t)$. Then, the ACS decides with a probability of p_x whether to choose an action randomly or to choose the best action. The best action is the action of the classifier with the maximum strength $q*r$ in the M_{set}. Next, an action set A_{set} is formed out of all classifiers in the M_{set} that propose the chosen action. After the execution of the action, first the ALP (with respect to the resulting state $\sigma(t+1)$) and then the GA modify the A_{set} and produce the learning set. Finally, the reinforcement component updates the reward measure r considering the perceived payoff $\rho(t)$ and the next match set $M_{set}(t+1)$. Figure 1 visualizes the process.

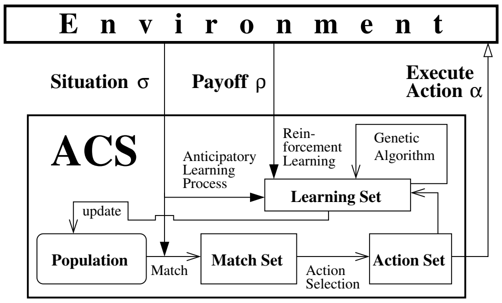


Figure 1: The figure visualizes the learning cycle in the ACS with all the involved learning procedures as well as the interaction with the environment.

2.2 THE ALP

The ALP modifies the action set with respect to the resulting state $\sigma(t+1)$. In the ALP new classifiers are formed out of inaccurate more general ones considering the mark M, the anticipation of the classifier, and $\sigma(t+1)$. Furthermore, the quality is updated using the Widrow-Hoff delta rule (Widrow & Hoff, 1960) with learning rate b_q (i.e. increase: $q = (1-b_q)*q+b_q$ and decrease: $q = (1-b_q)*q$). A more detailed description of the ALP can be found in Butz, Goldberg, and Stolzmann (2000).

2.3 THE GA

After the ALP modifies the learning set, the GA is applied. It is only applied if the average time since the last GA in the set is greater than the threshold θ_{ga}. Two classifiers are then selected with roulette wheel selection where the bid of each classifier is the cube of its quality (q^3). With a probability μ a specialized attribute in the condition part of the selected classifiers is mutated (i.e. generalized to a '#'-symbol). With a probability χ the two classifiers are crossed. The resulting classifiers are inserted into the population if they are not completely general. If the produced classifier already exists, its numerosity is increased. Finally, classifiers are deleted in the action set, if the size of the resulting set is greater than the action set size threshold θ_{as}.

2.4 REINFORCEMENT LEARNING

In order to realize reinforcement learning, an algorithm similar to the bucket brigade idea (Holland, 1985) and the Q-learning algorithm updates the r values of the classifiers in the resulting learning set.
$r = r + b_r * (\gamma * max_{cl \in M_{set}(t+1)}(q_{cl} * r_{cl}) + \rho(t) - r)$
Where b_r is the learning rate and γ the discount factor.

3 THE ENHANCEMENT OF THE ALP

As explained in Butz, Goldberg, and Stolzmann (2000), originally the ALP only modified the executed classifier in the action set. In order to balance the specialization strength, due to the ALP, with the generalization strength, due to the GA, and in order to learn as much as possible in one step the ALP was enhanced to work in the whole action set.

We tested the enhancement of the ALP in two mazes in order to show the resulting performance of the ACS. Figure 2 shows the Woods1 environment that was introduced by Wilson (1994) and the Maze4 environment which was investigated by Lanzi (1997) with the XCS classifier system. In all the presented maze experiments the ACS perceives the eight adjacent cells. A cell is either empty (.), or contains an obstacle (O) or food (F), thus $\sigma \in \{., O, F\}^8$. For instance the animat indicated in Woods1 as a '*' perceives $\sigma(t)$ ='.....OOF'. The ACS is able to move to these eight cells (if the cell is not blocked by an obstacle), which are the only possible actions in the environment(i.e. $\alpha \in \{N, E, S, W, NE, SE, SW, NW\}$). If the ACS reaches the food position, then the ACS receives a reward ρ and one trial ends (i.e. the ACS is reset to a random empty position). However, in the presented latent learning tests $\rho = 0$. The exploration ratio is set to $p_x = 0.8$. The learning rate in the ALP is $b_q = 0.05$ All presented results are averaged over 10 experiments.

O	O	F		
O	O	O	*	
O	O	O		

O	O	O	O	O	O	O	O
O			O			F	O
O	O			O			O
O	O		O			O	O
O							O
O	O		O				O
O					O		O
O	O	O	O	O	O	O	O

Figure 2: In the Woods1 (left hand side) and Maze4 (right hand side) environments the capabilities of the enhanced ALP and the GA are investigated.

Figure 3 shows the speed-up in the learning process of the ACS when enhancing the ALP application to the whole action set. Note that the size of the population is actually increasing when enhancing the ALP application. However, the huge speed-up compensates the decrease in speed due to the larger population.

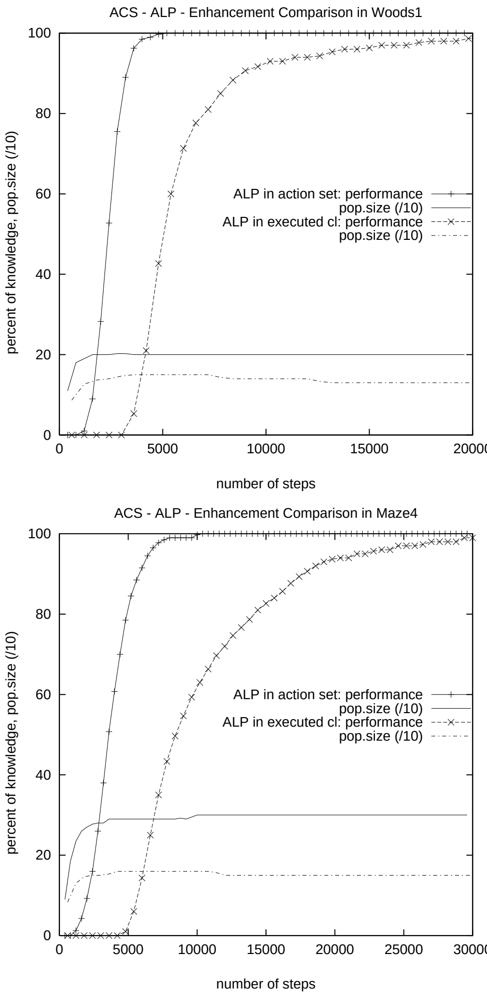


Figure 3: The enhancement of the ALP-application results in a huge learning speed-up. Compared to that, the population increases only slightly.

4 PERFORMANCE WITH GA

Butz, Goldberg, and Stolzmann (2000) introduced the GA to the ACS. The approach showed the good performance of the GA in a simple maze, the Woods1 environment (Figure 2). First, this section further analyzes the generalization capability of the ACS with the GA in Woods1 further. Next, it examines more challenging environments. If not stated otherwise, the parameters are set to $b_q = b_r = 0.05$, $\gamma = 0.95$, and $p_x = 0.8$. The GA parameters are set to $\theta_{ga} = 100$, $\mu = 0.3$, $\chi = 0.0$, and $\theta_{as} = 20$. The payoff $\rho = 1000$ when the food position is reached (in the maze environments) or a correct action was executed (in the multiplexer environment), and $\rho = 0$ otherwise. However, in the latent learning experiments ρ is always 0. All presented results are again averaged over ten experiments.

4.1 GENERALIZATION IN WOODS1

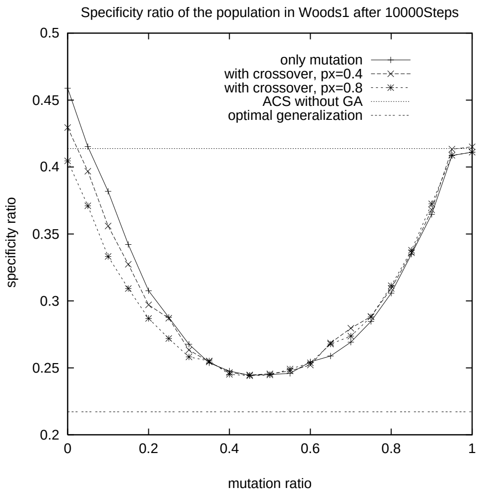
To be able to analyze the generalization in the population of the ACS we introduce the specificity measure $spec(pop)$. It is equal to the sum of all specialized attributes in C of all classifiers in the population divided by the number of classifiers times the length of σ. Figure 4 analyzes the amount of generalization for different mutation and crossover values after 10000 executed steps in Woods1. The result shows that at $\mu = 0.4$ the best generalization is reached. Interesting is the fact that the GA never causes any over-generalization. This indicates that the ALP is strong enough to conquer the genetic generalization pressure and is even adapting to different amounts of pressure. When μ is greater than 0.5, mutation increasingly produces completely general classifiers which are not inserted into the population, and thus the genetic generalization pressure decreases.

Crossover appears to have no positive effect on the generalization capability. The mixing due to crossover does not help to create more general classifiers more quickly. The performance comparison in Figure 4 shows that crossover does not help to increase the performance of the ACS, either. Thus, we decided not to use crossover in the further presented results. Runs with crossover in the other environments (not presented) resulted in similar performances, specificities, and population sizes. Sometimes it even appeared to be disrupting. This is certainly due to the properties in the presented maze environments. Each position has a similar probability of importance. Interactions can occur in neighboring attributes as well as more distant attributes. This may result in long defining lengths of building blocks (Goldberg, 1989) which explains the disruption due to crossover.

4.2 GENERALIZATION OF IRRELEVANT ATTRIBUTES

The specificity ratio in the Woods1 experiment shows that the ACS with the GA is now able to generate much more general classifiers than before. But, the classifier list actually reveals a temporary increase in size and only later in the run convergence to a small size. Thus, it is not clear why the GA is any good. We want to illustrate the usefulness of the GA on the following example.

Considering a real robot that is equipped with different kinds of sensors, perceptions are normally not as clear as in the maze environments. For example, the



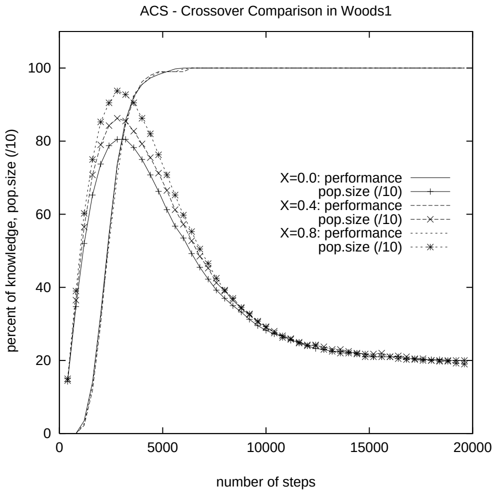


Figure 4: In Woods1, the best generalization is achieved, when setting μ to 40%. Crossover has neither a positive effect on the specificity, nor on the performance in Woods1.

robot could detect different degrees of brightness in its surrounding. Therefore, the trials in one experiment can occur under different light conditions. For a system without any generalization capabilities this would result in a new environmental perception, each time the light condition changes. Here we show that the ACS is able to handle such irrelevant attributes.

In order to simulate such a light scenario we introduce three further attributes into the perception string that can be placed at any position. These three attributes are always randomly changed between zero and one when a new trial starts. Figure 5 shows the performance of the ACS without and with the GA under such conditions in Maze4. We can now observe that the ACS without the GA is able to handle the irrelevant attributes, too, but the ACS with the GA is doing better. The complete cognitive map is learned faster and the resulting population is smaller. This shows for the first time that the ACS is capable of forming the admirable attention spot (Butz, Goldberg, & Stolzmann, 2000); v.i.z. the irrelevant light source is not considered anymore.

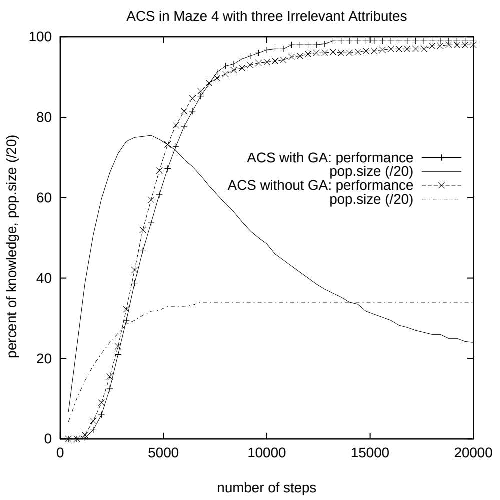


Figure 5: When adding three irrelevant attributes into the perceptions in Maze4, the ACS with GA is beating the ACS without GA in the performance and the size of the population later in the run.

The presented results showed so far that the size of the population of the ACS with GA is at first increasing faster than without GA. However, the size approached the size of the population without GA in the end of a run. In the light example, the size even got smaller. In the next section we investigate this property of convergence further.

4.3 CONVERGENCE IN THE MULTIPLEXER ENVIRONMENT

In order to study the convergence in the ACS further we apply it in the Multiplexer environment. Wilson (1995) and Wilson (1998) tested XCS in this environment in order to test the generalization capabilities.

The multiplexer environment generates a random bit string of length l. The string is evaluated by the corresponding multiplexer function which is defined for length $l = k + 2^k$ ($k \in IN$) where the first k bits address one bit in the 2^k remaining bits. The function returns the addressed bit. In order to make this environment solvable for the ACS we need to introduce a perceptual causality into the environment (Butz, Goldberg, & Stolzmann, 1999). This can be done by adding another attribute to the string (thus, $L = l+1$) which is 0 in each problem instance and switches to 2 if the correct action was executed and to 1 otherwise. If the correct action was executed, the ACS receives a reward $\rho = 1000$. After that, one trial ends and the environment generates another random string.

As in the XCS application, we divide the learning into one pure exploration and one pure exploitation step. In the exploration mode the action is chosen as before and all the learning mechanisms are present. In the exploitation mode neither the ALP, nor the GA, nor the reinforcement mechanism is active. Here, we consider the 11-multiplexer function in which the first three bits address the remaining eight bits. For example, the correct action in the situation $\sigma(t)$ ='010110111110' is 0 since the first three bits ('010') address the bit number two in the eight remaining bits. If the correct action was executed, the resulting state would be '010110111112'. Figure 6 shows the performance of the ACS without and with GA in this environment. Without the GA, the ACS is not able to generalize and thus produces much more classifiers than with GA. The specificity ratio (not shown in the graph) is equal to 0.83 after 15000 exploration problems. This indicates that the environment was mainly learned by representing the perceptions without any generalization. With the GA the ACS is generating more general classifiers (spec(pop)=0.57 after 15000 explore problems) and is thus able to reach an optimal performance faster. Furthermore, the size of the population is much smaller than the one in the ACS without GA

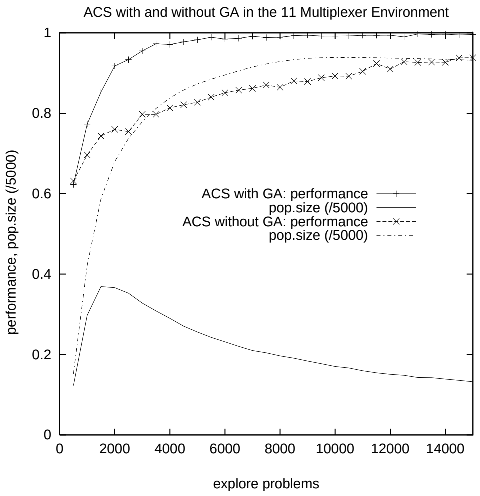


Figure 6: When adding the GA, the ACS is able to learn the multiplexer task correctly and the population converges to fewer classifiers.

Figure 7 compares the performance of the ACS with the performance of XCS. The implementation of XCS applies subsumption deletion (Wilson, 1998) and the constants are set equal to Wilson (1995). In this comparison we can observe that the ACS is reaching a near optimal performance faster than XCS. But the price is a much larger population. The population size curve of the ACS is basically similar to the one of XCS but is situated at a much higher level. The population is converging but not yet as well as in the XCS classifier system. An investigation of the classifier list showed that one problem is classifiers that are completely specific and thus only occur very seldom in an action set. These classifiers are difficult to handle for the GA as they are only very seldom part of the GA process. Since the deletion method in the ACS works on the action set and not on the whole population, as does XCS, the probability of deleting such a specific classifier is very small. Despite that, the comparison shows that the ACS – although actually invented for other applications – is able to solve the multiplexer problem with a performance similar to XCS.

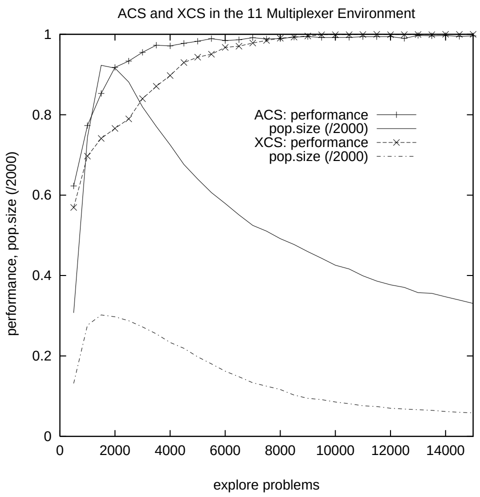


Figure 7: Comparing the ACS with GA to XCS, we can observe that the ACS is reaching a near optimal performance faster than XCS. However, XCS needs fewer classifiers to solve the task and converges to a lower level.

4.4 REINFORCEMENT LEARNING IN DIFFERENT MAZES

Up to now, we showed that the ACS with GA is indeed generating more general classifiers and the population is converging to a smaller size. Moreover, we were able to show the usefulness of the process in a maze with additional irrelevant attributes and in the Multiplexer problem. What remains is to investigate the reinforcement mechanism.

O	O	O	O	O	O	O	O	O
O						O	F	O
O			O		O	O		O
O		O						O
O				O	O			O
O		O		O			O	O
O		O						O
O						O		O
O	O	O	O	O	O	O	O	O

Figure 8: The Maze6 is a challenging environment because of the different frequencies in experiencing different states.

4.4.1 XCS In Different Mazes

The first published XCS performances in environments with deferred reward (Woods1 and Woods2) did not reveal any problems with the reinforcement learning mechanism. However, these environments were regular, aperiodic, and many generalizations were possible. Lanzi (1997) investigated the generalization mechanism of XCS in a more challenging environment: Maze4 (Figure 2). It was observed that XCS is misled by inaccurate overly general classifiers. The problem got even worse in the Maze5 and Maze6 environments (Maze6: Figure 8, Maze5 is similar) that were investigated by Lanzi (1999). The introduced specify mechanism in XCS was able to overcome the problem. Moreover, the introduction of a tele-transportation mechanism showed that the problem occurs because of the different frequencies of experiencing the different environmental niches. Positions near to the food are experienced much more often than positions which are more distant. Therefore, XCS was sometimes unable to get rid of overly general classifiers fast enough. The question is whether the ACS also suffers from this difference in the frequencies and if so, to what degree it suffers.

4.4.2 The ACS in Maze6

Figure 9 shows that the ACS is converging to an optimal solution. Like XCS, the ACS alternates between one trial of exploration and one trial of pure exploitation. The performance of the trials with pure exploitation are presented. Over-generalization problems similar to XCS cannot be observed. As the learning is based on the anticipations and not on the perceived reward, the ACS does not experience any problems. The same results were achieved for the Maze5 environment, which appeared to be simpler for XCS. The resulting size of the population is now even smaller than the one in XCST without subsumption (Lanzi, 1999)[page 147] and only slightly larger than the size in XCST with subsumption.

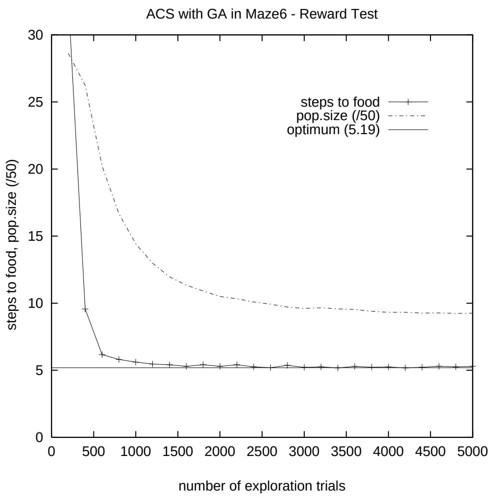


Figure 9: The ACS does not have any problems solving the reinforcement learning task in Maze6.

4.4.3 The ACS in Woods14

A real challenge for the reinforcement mechanism can be observed in the Woods14 environment (Figure 10). Cliff and Ross (1994) examined the performance of the ZCS (Wilson, 1994) and revealed the problem of forming long chains with the bucket brigade. The same problem was observed in XCS and now in the ACS. When leaving the learning parameters unchanged an optimal performance cannot be reached.

O	O	O	O	O	O	O	O	O	O	O	O	O
O	O				O	O	O	O		O	O	
O		O	O	O		O	O		O		O	
O		O	O	O		O		O	O	O		O
O	F	O	O	O		O	O		O	O	O	O
O	O	O	O	O	O			O	O	O	O	O

Figure 10: Woods14 especially challenges the reinforcement learning algorithm as the reward needs to be back-propagated over many steps.

Figure 11 shows that this depends mainly on the discount factor γ as proposed in Cliff and Ross (1994). Moreover, an optimal performance cannot be reached when applying pure exploitation (i.e. $p_x = 1.0$) as observed in XCS as well (Lanzi, 1999). This is mainly the case because otherwise the ACS tends to walk into the obstacles and the reward is not back-propagated appropriately. However, a problem of over-generalization as in XCS could not be observed in the ACS. Due to the anticipation-learning, the dependence on the correct reward-distribution decreases and thus the correct policy is learned more easily.

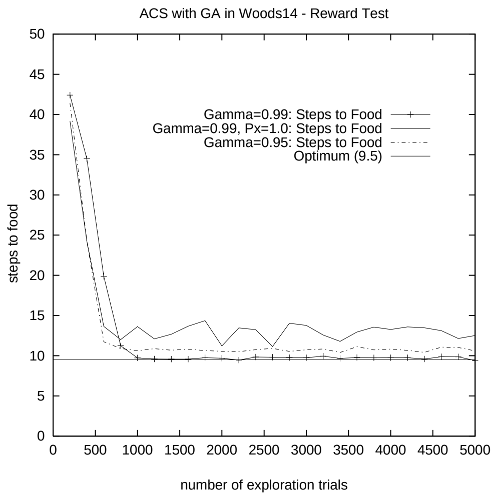


Figure 11: In order to solve this task with the ACS the parameters need to be modified, so that the reward can be back-propagated farther.

5 CONCLUSION

The study of the genetic algorithm in the ACS showed that the ACS with GA is able to generate accurate, maximally general rules. Once these rules are generated, the size of the population decreases and converges at a low level. Moreover, an investigation of the reinforcement learning capabilities showed that the GA does not disrupt the reinforcement learning mechanism. The learning rates showed that the ACS performance is comparable to the XCS performance in similar environments. However, in addition to the payoff map, the ACS generates a cognitive map of the environment.

Up to now, the evolving cognitive map in the ACS was not used to increase the reinforcement learning capabilities. For example, the mental representation of the environment could be used to execute hypothetical actions similar to Dyna (Sutton, 1991). This should result in far fewer necessary actual trials to reach optimal performance as the reinforcement learning will mainly be done mentally. Moreover, a more directed exploration of environmental niches could be performed. Due to the existing cognitive map, the ACS should be able to observe where its knowledge is not completely reliable and thus could actively explore such environmental niches. Future research will show what kind of cognitive processes are possible in the ACS and where the system is limited.

Acknowledgments

The authors would like to thank Dimitri Knjazew, Martin Pelikan, and Stewart Wilson for valuable discussions and useful comments. The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-97-1-0050. Research funding for this work was also provided by the National Science Foundation under grant DMI-9908252. Support was also provided by a grant from the U. S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0003. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. Additionally, research funding was provided by the German Research Foundation DFG. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, the U. S. Army, or the U. S. Government.

References

Booker, L. B., Goldberg, D. E., & Holland, J. H. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence*, *40*, 235–282.

Butz, M. V., Goldberg, D. E., & Stolzmann, W. (1999). *New challenges for an Anticipatory Classifier System: Hard problems and possible solutions* (IlliGAL report 99019). University of Illinois at Urbana-Champaign: Illinois Genetic Algorithms Laboratory.

Butz, M. V., Goldberg, D. E., & Stolzmann, W. (2000). Introducing a genetic generalization pressure to the anticipatory classifier system - Part 1: Theoretical Approach. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. In press.

Cliff, D., & Ross, S. (1994). Adding Temporary memory to ZCS. *Adaptive Behavior*, *3*(2), 101–150.

Drescher, G. L. (1991). *Made-Up Minds, a constructivist approach to artificial intelligence.* Cambridge, MA: MIT Press.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning.* Reading, Massachusetts: Addison-Wesley.

Hoffmann, J. (1993). *Vorhersage und Erkenntnis [Anticipation and Cognition].* Goettingen, Germany: Hogrefe.

Holland, J. H. (1985). Properties of the bucket brigade algorithm. In *Proceedings of an international conference on genetic algorithms and their applications* (pp. 1–7). Carnegie-Mellon University, Pittsburgh, PA: John J. Grefenstette.

Lanzi, P. L. (1997). A study of the generalization capabilities of XCS. In Baeck, T. (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithm* (pp. 418–425). San Francisco, California: Morgan Kaufmann.

Lanzi, P. L. (1999). An analysis of generalization in the XCS classifier system. *Evolutionary Computation*, *7*(2), 125–149.

Riolo, R. L. (1991). Lookahead planning and latent learning in a classifier system. In Meyer, J.-A., & Wilson, S. W. (Eds.), *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* (pp. 316–326). Cambridge, MA: MIT Press.

Stolzmann, W. (1997). *Antizipative Classifier Systems [Anticipatory Classifier Systems].* Osnabrueck, Germany: Shaker Verlag, Aachen, Germany.

Stolzmann, W. (1999). Latent learning in Khepera robots with Anticipatory Classifier Systems. In *2.International Workshop on Learning Classifier Systems (2.IWLCS) on the Genetic and Evolutionary Computation Conference (GECCO-99)* (pp. 290–297). Orlando, Florida: Morgan Kaufmann.

Sutton, R. S. (1991). Reinforcement learning architectures for animats. In Meyer, J.-A. (Ed.), *Proceedings of the first international conference on simulation of adaptive behavior* (pp. 288–296).

Widrow, B., & Hoff, M. (1960). Adaptive switching circuits. *Western Electronic Show and Convention*, *4*, 96–104.

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

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, *3*(2), 149–175.

Wilson, S. W. (1998). Generalization in the XCS classifier system. In Koza, J. R. e. a. (Ed.), *Genetic Programming 1998: Proceedings of the third annual conference* (pp. 665–674). San Francisco: Morgan Kaufmann.