Introducing a Genetic Generalization Pressure to the Anticipatory Classifier System - Part 1: Theoretical approach

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 a learning classifier system that is based on the cognitive mechanism of anticipatory behavioral control. Besides the common reward learning, the ACS is able to learn latently (i.e. to learn without getting any reward) which is not possible with reinforcement learning techniques. Furthermore, it forms a complete internal representation of the environment and thus is able to use cognitive processes such as reasoning and planning. Latest research observed that the ACS is not generating accurate, maximally general rules reliably (i.e. rules which are accurate and also as general as possible), but it is sometimes generating over-specialized rules. This paper shows how a genetic algorithm can be used to overcome this present pressure of over-specialization in the ACS mechanism with a genetic generalization pressure. The ACS works then as a hybrid which learns latently, forms a cognitive map, and evolves accurate, maximally general rules.

1 INTRODUCTION

Recently, research has payed a growing attention to learning classifier systems (LCSs). They were successfully applied in various classification tasks such as function and data classifications but also tasks with deferred reward (e.g. different types of mazes). Most notably the XCS classifier system, which was developed by Wilson (1995), proved the capabilities of LCSs.

The Anticipatory Classifier System (ACS) is an enhanced type of LCS which is able to form an internal model of the environment (i.e. a cognitive map). It was introduced by Stolzmann (1997). Besides the common reward learning abilities of LCS, the ACS is able to learn latently (Stolzmann, 1998). It autonomously explores an environment and forms a cognitive map without perceiving any reward. The ACS is able to use the evolving cognitive map for doing lookahead planning as presented in Stolzmann (1999) or action planning (Butz & Stolzmann, 1999). The use of other cognitive processes still needs more investigation. The ACS could e.g. use the cognitive map to learn from hypothetical experiences similar to the Dyna Architecture (Sutton, 1991). The advantage of the ACS is that it is not forming an identical internal image of the environment (like Dyna) but the environmental representation is more generalized. However, up to now the ACS did not use any type of generalization in the learning process but only carefully specialized rules from more general ones using the anticipatory learning process (ALP). Butz, Goldberg, and Stolzmann (1999) showed that despite the careful specialization process sometimes over-specialization occurs.

This paper introduces a genetic generalization pressure to the ACS. It shows that the generalization pressure can be integrated in the ACS in order to generate accurate, maximally general classifiers while not interrupting the formation of a cognitive map and the latent learning ability of the ACS. Also, it shows that it is possible to combine an indirect genetic generalization with a directed specialization in LCSs.

The next section gives an introduction to the ACS. After that, Section 3 explains the ALP in more detail, enhances the application of the process, and especially reveals the pressure of specialization and the causes of over-specialization in the process. Section 4 introduces the new generalization pressure realized by a genetic algorithm and Section 5 clarifies the interaction of the two pressures. Section 6 tests the process on a first simple task. Finally, a summary is given.

2 OVERVIEW OF THE ACS

Stolzmann (1997) introduced the basic ACS. It is based on the cognitive mechanism of anticipatory behavioral control (Hoffmann, 1993) and on learning classifier systems (Booker, Goldberg, & Holland, 1989). First applications in Stolzmann (1998) showed the reliability of the cognitive mechanism and the capability of learning latently in addition to the common reward learning ability in LCSs. Stolzmann (1999) revealed that the ACS is generating a complete cognitive map of the perceived environment and is able to do action planning. In a hand-eye coordination task, Butz and Stolzmann (1999) showed that the ACS is able to use the ability of action planning during learning to build a complete cognitive map of the environment much faster. The remainder of this section gives an overview of the basic structure and mechanism in the ACS. For a more complete overview of the ACS the interested reader should refer to the cited papers.

2.1 FRAMEWORK

Like all LCSs, the ACS interacts with a certain environment. After the execution of an action, it is able to do reward learning. But the main learning mechanism is the ALP which learns latently by comparing its own anticipations with the perceptual consequences (i.e. the resulting state). Therefore, to be able to learn, an environment must have a perceptual causality in its successive states (Butz, Goldberg, & Stolzmann, 1999).

The perceived states $\sigma(t)$ are coded as strings. Each attribute does not need to be binary but can only take discrete values. Thus, a state $\sigma(t) \in \{s_1, s_2, ..., s_m\}^L$, where L is the length of the strings, m is the number of different possible values, and $s_1, ..., s_m$ are the different values. The possible actions $\alpha(t)$ are also coded with discrete values (i.e. $\alpha(t) \in \{r_1, r_2, ..., r_n\}$ where n is the number of different actions and $r_1, ..., r_n$ are the different actions). The reward $\rho(t)$ is a simple real value.

2.2 STRUCTURE

The ACS knowledge is represented in its classifier list (i.e. its population). Each classifier has the following parts:

- A *condition part (C)* specifies the conditions for being applied in an environmental state.
- An *action part (A)* specifies the action the classifier will execute, once applied.
- An *effect part (E)* anticipates the perceptual consequences in the environment after the execution of an action.
- A *mark (M)* records all different values of each attribute in the states where the classifier did not anticipate correctly. Thus, the mark serves as a reminder that at times, the classifier does not work correctly, as well as gives information about how to differentiate the classifier from the incorrect states when applied in a correct state.

Furthermore, each classifier has a quality measure $q \in [0, 1]$ which reflects the accuracy of its anticipation and a reward measure $r \in IR$ which predicts the expected payoff after the execution of an action.

The condition and the effect part consist of the values perceived from the environment and of '#'-symbols (i.e. $C, E \in \{s_1, ..., s_m, \#\}^L$). A '#'-symbol in an attribute of the condition part, which is called 'don't-care' symbol, matches any perceived value in this attribute. The more 'don't-care' symbols are in the condition part, the more general is a classifier. A '#'-symbol in an attribute of the effect part, which is called 'pass-through' symbol, predicts that this attribute will remain the same (i.e. the corresponding value in the perception does not change after the execution of an action). The action part consists of any action which is possible in the environment (i.e. $A \in \{r_1, ..., r_n\}$). The mark has the structure $M = (m_1, ..., m_L)$ with $m_i \subseteq \{s_1, ..., s_m\}$.

2.3 A BEHAVIORAL ACT

A behavioral act starts by forming a match set out of the current population with respect to the current perception $\sigma(t)$. Out of this match set a specific action is chosen. The action is chosen either randomly with a probability of p_x to increase exploration or the best action is chosen (i.e. the classifier with the highest strength, where strength is defined as $q * r$). The chosen action is then executed and the resulting action set (i.e. all classifiers in the match set that propose the chosen action) is modified by comparing the perceptual consequences $\sigma(t+1)$ with the made anticipation. After the modification, the next match set is formed with the new perception $\sigma(t+1)$. Figure 1 visualizes this process.

The ALP, performed by the above modification process, is based on the theory of anticipatory behavioral control. This process generates more specialized classifiers out of more general ones. The next section clarifies the process in more detail and enhances its application.

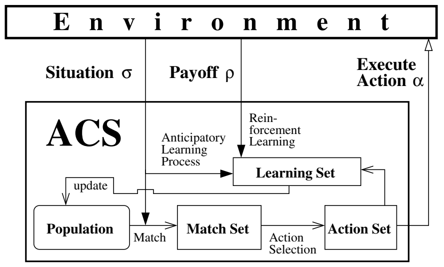

Figure 1: The ACS uses its own anticipations and the consecutive perceptions to form new classifiers out of the classifiers which match the current perception and demand the chosen action.

3 SPECIALIZATION PRESSURE

The ACS increases its knowledge in each behavioral act by modifying its population. It uses the ALP to do so. The process compares the anticipation of a classifier with the perceived consequences after the execution of the corresponding action and learns by the differences. It generates more specialized classifiers out of inaccurate, more general ones. It does not generalize any classifiers. The process resembles a specialization pressure in the ACS. The remainder of this section clarifies the ALP, at first. Next, the application of the process is enhanced. Finally, the involved pressure of specialization and the causes for over-specialization are revealed.

3.1 THE ANTICIPATORY LEARNING PROCESS (ALP)

The ACS starts with a population that contains only completely general classifiers (i.e. classifiers, whose C and E part contain only '#'-symbols), one for each possible action. In one behavioral act the action of one classifier is executed and the ALP is applied once. At first, the process specializes the *changing attributes* in the C and E part. After that, in order to differentiate such a modified classifier from states in which the changes do not take place, the process specializes *unchanging attributes* in the C part. Four cases realize the ALP:

- In the *correctable case* a classifier does not correctly anticipate the consequent state $\sigma(t+1)$ but it can be further specialized (i.e. '#'-symbols can be changed to specific values). In this case, the process marks the classifier by the state $\sigma(t)$ and the quality q is decreased. Furthermore, it generates a new classifier and specializes it in the changing attributes. Therefore, the new classifier will now anticipate correctly when applied in this state.
- In the *not correctable case* a classifier does not anticipate correctly but cannot be further specialized (i.e. one or more attributes that did the wrong prediction are already specialized in the E part). In this case, the process marks the classifier by $\sigma(t)$ and decreases q.
- In the *useless case* a classifier predicts changes, but actually no changes occur in the environment. Also here, the process marks the classifier by $\sigma(t)$ and decreases q.
- In the *expected case* a classifier anticipates correctly. Now, the process increases q. Furthermore, if the classifier is marked, it generates a new classifier and differentiates it from the states stored in the mark by comparing the attributes in the mark with the attributes of the state $\sigma(t)$.

q is increased and decreased by 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: $q = (1 - b_q) * q$).

3.2 MODIFICATION OF THE WHOLE ACTION SET

So far, the ACS executed and modified one classifier in each behavioral act according to the ALP. Analysis of the running time of various components in a behavioral act revealed that the ACS needs the highest amount of time to form the match set out of the population and the current perception. This observation suggests that the ACS should learn as much as possible out of a given match set. The amount of learning in one behavioral act can be increased by applying the learning process to the whole action set rather than to only one classifier in the action set.

The application of the ALP in the whole action set proceeds as follows:

- Each classifier in the action set undergoes the ALP.
- A new classifier that exists already is not created but the quality q of the identical classifier is increased.
- After all the classifiers in the action set are processed, the new classifiers are added to the population.

Experiments in various environments have shown that this enhancement does not interfere with the ACS mechanism but increases speed and tremendously decreases the number of learning steps in all cases. Reward learning tests in different mazes show similar good results. In Butz, Goldberg, and Stolzmann (2000) the better performance is illustrated on two examples.

3.3 CAUSES FOR OVER-SPECIALIZATION

Although published results (Stolzmann, 1999, Butz & Stolzmann, 1999) show that the ALP builds a complete cognitive map and the classifier list stays reasonably small, the formed classifiers are not necessarily maximally general. There are three reasons why a classifier is not maximally general in the condition part C:

- Changing attributes are always specialized in C. However, sometimes the specialization in C is not necessary because there can exist another component that uniquely specifies the state or only a part of the changing components would be sufficient to uniquely specify the state.
- Furthermore, the mark can give wrong or incomplete information.
  - The classifier could have experienced only a part of the states where it does not anticipate correctly. Therefore the mark can be incomplete and it is not necessarily clear which component is the best one to specialize.
  - The structure of the mark itself is limited. The mark does not store all states where the classifier did not anticipate correctly, but it does store all different values of each attribute of all states in which it did not anticipate correctly. Therefore important interactions between different unsuccessful states can get lost.

  Thus, when specializing unchanging components, sometimes unnecessary components will be specialized.
- Finally, the marks can give wrong information in nondeterministic environments which is investigated in somewhat more detail in Butz, Goldberg, and Stolzmann (1999).

If all possible states where a classifier does not work correctly were stored in the mark, the size of the mark would increase tremendously. Therefore, the structure of the mark should not be changed. Furthermore, the ALP showed that it is able to solve suitable problems quickly, accurately and reliably and should therefore not be further modified. The next section shows how the bias on over-specialization can be overcome by introducing a generalization pressure that is realized by a genetic algorithm (GA).

4 GENERALIZATION PRESSURE

As shown in Section 3.3 the ACS sometimes generates over-specialized classifiers. The reasons for the over-specialization revealed no possible enhancement of the ALP. Therefore, the pressure of over-specialization will be alleviated by a genetic generalization pressure, which will be realized by a simple GA. Before we explain the functioning of the GA, we will first show why it is very useful in the ACS to be able to generate accurate, maximally general rules. Next, we will lead to the idea of the GA application.

4.1 ACCURATE, MAXIMALLY GENERAL RULES IN THE ACS

In psychology the different mechanisms of attention in animals and humans are intensively investigated. The visual spot is the simplest example of an omnipresent attention mechanism in ourselves. Paying attention to too many things at the same time is disturbing and increasingly impossible. Infants are able to distinguish things and pay attention to something while ignoring other things in the earliest stages of their development. Thus, considering the ACS as a cognitive system which is supposed to be able to learn in real worlds, it is clear that an identical internal representation of the world is not an option. The amount of information available in our world is just too huge to be able to consider all of it at the same time.

But also in a more abstract view the necessity of maximally general rules can be revealed. Let us simply consider an environment where one attribute is irrelevant in the perceived strings but in some learning trials it is one and in others it is zero (e.g. a robot is supposed to learn a maze and perceives different light conditions). Such a simple attribute would double the search space in a system that learns completely specialized rules and thus significantly decrease the speed of cognitive processes, especially learning.

4.2 THE GENERAL IDEA FOR THE GA

The general idea for the GA is that it should introduce a generalization pressure to create maximally general classifiers out of the over-specialized classifiers generated by the ALP. While generating the classifiers, it should not interfere with the ALP. It should neither change the learning of the effects of an action nor destroy the structure of the conditions or over-generalize the conditions.

To achieve this goal, the GA takes many ideas from the XCS classifier system and applies them to the ACS.

XCS uses a measure of the accuracy of its reward prediction to learn. By using this accuracy measure, XCS can learn an environment more reliably and completely than other LCSs. Indeed, it is able to form a complete payoff map of the environment. The capability of generating accurate, maximally general classifiers was mentioned in Wilson (1995) and was then further investigated and enhanced in Wilson (1998). Kovacs (1996) enhanced the generalization hypothesis in Wilson (1995) to an optimality hypothesis, which suggests that XCS is eventually evolving an accurate, maximally general classifier for any payoff level which it is able to sample sufficiently.

The quality measure q of a classifier in the ACS reflects the accuracy of the anticipations of a classifier. Due to the similarities to the reward accuracy measure in XCS, this measure is used as the *fitness measure*, which will act similarly to the fitness in XCS. Furthermore, as in XCS each classifier carries an additional *time stamp* that specifies the last time the classifier was in a set where the GA took place. The time stamp is used to control the frequency of the GA application. The deletion method in XCS is executed upon the population. It takes an estimate of the action set size of each classifier into account and deletes classifiers proportional to that size in order to ensure an equal amount of classifiers in each environmental niche. (Under certain conditions the fitness is also considered.) This mechanism is modified so that each classifier does not need to keep an action set size estimate, but there is a *fixed action set size* (rather than a fixed population size in XCS). The deletion is realized inside an action set considering the fitness as well as the specificity of a classifier.

4.3 HOW THE GA WORKS

The GA mechanism works as follows:

- The GA is applied in the action set after the ALP.
- After the ALP the ACS determines in the modified action set if the GA should be applied. (i.e. the average time of the classifiers in the action set since the last GA, which is calculated by the time stamp, must be bigger than a certain threshold θ_{ga}).
- If the GA is applied, two classifiers are selected by roulette wheel selection. The contribution to the roulette wheel selection of each classifier is determined by the cube of the quality of a classifier (i.e. q^3). The cube is used in order to increase the probability of reproducing more accurate classifiers.
- The parents stay in the population.
- With a probability μ, each specialized attribute in the conditions is mutated (i.e. changed to a 'don't care' sign).
- With a probability χ, the two classifiers are combined by one-point crossover.
- The resulting classifiers are inserted into the population only if they are not completely general.
- Identical classifiers are now allowed in the population. Therefore, each classifier keeps an additional counter (i.e. the numerosity *num*) of how many classifiers it represents, which is similar to the macro-classifiers in XCS (Wilson, 1995).
- If the action set size exceeds the action set size threshold θ_{as}, excess classifiers are deleted.
- The deletion method deletes classifiers using tournament selection with a tournament size of $\frac{1}{3}$ the action set size. The tournament is held in two stages. At first the quality is considered - the worse classifier gets deleted. But, if the qualities are approximately the same (within a range of 0.1), then the more specialized classifier is deleted.

The cube of the quality q in the selection method is right now chosen arbitrarily. In XCS the fitness is calculated in a more sophisticated way. In contrast to the ACS, XCS first calculates an error of its reward anticipations. Using this error, it then determines the accuracy of the classifier with a modified exponential function. After that, the update of the fitness of a classifier is calculated out of the accuracy, relative to the accuracies of the other classifiers in the set (Wilson, 1995). As the ACS simply takes the quality measure q for its fitness, this measure needs to be modified to further bias the selection pressure on more accurate classifiers. The chosen cube function resulted in the best performance so far.

The one-point crossover chooses the crossing site randomly. Right now the crossing works only in the C part of a classifier. An enhancement of the crossover to work also in the effect part is imaginable. But, as explained earlier, the effects are correctly generated by the ALP. Thus, the structure of the E part could be destroyed by crossing this part. Moreover, the aim of the GA is to generalize the conditions and not to find new possible effects of an action.

The deletion method relies on two factors. θ_{as} must be big enough so that the GA possibly works but on the other hand the bigger it is the longer it takes to converge to the necessary classifiers. The tournament size of $\frac{1}{3}$ the action set size is chosen in order to keep a certain randomness in the deletion but also to increase the generalization pressure. A helpful analysis of tournament sizes in GAs can be found in Goldberg and Deb (1991).

The generalization pressure is realized by the mutation. Furthermore, the deletion method that rather chooses more specialized classifiers for deletion (if the classifiers have the same quality) increases the pressure. The next section reveals how the two pressures interact and presents one more enhancement in order to assure the convergence of the population to the accurate, maximally general rules.

5 UNIFICATION OF THE ALP AND THE GA

What remains is to unite the two processes in the ACS. The processes are supposed to work together and not against each other. While the ACS still should form a complete cognitive map, additionally now it is supposed to find the maximally general classifiers.

5.1 INTERACTION

The existing anticipatory specialization pressure, realized by the ALP, interacts with the introduced genetic generalization pressure, realized by the GA. The generalization pressure must neither be too strong nor too weak. If it is too strong, the population will become over-general and the ALP will not fix the problem fast enough. If it is too weak, the GA will not have any effect but simply disturb the ALP.

Figure 2 shows how the two pressures interact in the condition part of the classifiers. The ALP specializes the conditions. Both, the specialization of changing components and the specialization of unchanging components sometimes over-specialize and sometimes specialize perfectly. But, once a maximally general classifier is found it is not further specialized by the mechanism. An over-specialized classifier is not further specialized, either.

The GA generalizes the conditions. The offspring-selection prefers the most accurate classifiers which are those that are over-specific or maximally general. At first, the available classifiers will be those that are over-specialized by the ALP. If only maximally general classifiers exist some of those are over-generalized. But, the deletion method will delete those over-general classifiers fast, as they will become inaccurate.

5.2 INTRODUCING SUBSUMPTION

The interaction shows that the two processes already work in order to achieve the common goal. Indeed,

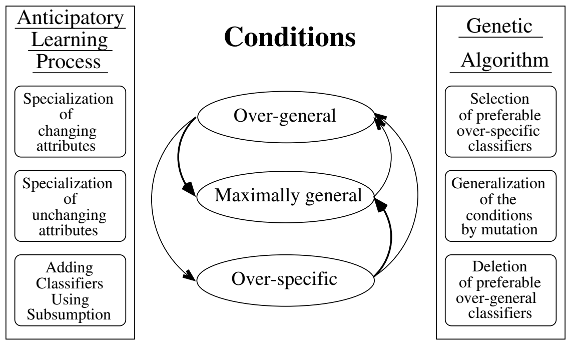
Anticipatory Learning Process
Specialization of changing attributes
Specialization of unchanging attributes
Adding Classifiers Using Subsumption
Conditions
Over-general
Maximally general
Over-specific
Genetic Algorithm
Selection of preferable over-specific classifiers
Generalization of the conditions by mutation
Deletion of preferable over-general classifiers

Figure 2: While the ALP realizes the specialization pressure, the GA generalizes. The different arrows in the graph show where the two pressures act.

first experiments showed that the ACS is now generating the desired accurate, maximally general classifiers. But what happens when the population is mainly converged? The ALP will continue to generate over-specific classifiers out of the over-general ones, which are generated by the GA and by the ALP itself out of the completely general classifiers. In order to achieve a more converged population there is a process necessary that stops the over-specialization. Thus, we use a subsumption method similar to the one introduced in Wilson (1998) in XCS.

When the ALP or the GA creates a new classifier cl_{new} the current action set is checked for a classifier cl_{sub} that subsumes cl_{new}. A classifier subsumes cl_{new} if it is (1) reliable (i.e. the quality is greater than 0.9), (2) experienced (i.e. it was applied more than 20 times), (3) not marked, (4) more general than cl_{new}, and (5) has the same action. When cl_{sub} subsumes cl_{new}, cl_{new} is abandoned. Furthermore, the quality q of cl_{sub} is increased when in the ALP, or the numerosity num is increased when in the GA. This method showed a further speed-up in the formation of the environmental representation as well as the convergence in the population.

The result is that the ALP, once the accurate, maximally general classifiers are found, is not generating over-specific rules anymore and the population can converge further.

6 PERFORMANCE OF THE ACS

This section compares the performance of the ACS without and with GA application in the Woods1 environment. It shows, that a much lower specificity ratio is achieved while the performance only decreases slightly. The parameters are set to the following values: The learning rate $b_q = 0.05$, the exploration ratio $p_x = 0.8$, the GA application threshold $\theta_{ga} = 100$, the mutation ratio $\mu = 0.3$, the crossover probability $\chi = 0.8$, and the action set size threshold $\theta_{as} = 20$. The mutation ratio μ is very high, but since in this case mutation is only generalizing and only a part of the attributes are specialized, this high ratio gives approximately optimal performance. A more detailed analysis of the mutation ratio is given in Butz, Goldberg, and Stolzmann (2000).

The Woods1 environment was first published in Wilson (1994). It is a two-dimensional grid environment with an object configuration that repeats indefinitely in the horizontal and vertical directions. Figure 3 shows the object configuration with an animat (the '*'). The animat perceives the eight nearest cells (i.e. '.....OOF') and can move in the eight directions (if the position is not blocked by an obstacle 'O').

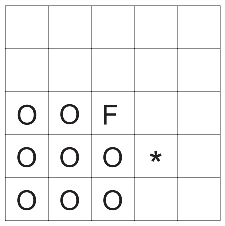

Figure 3: The Woods1 environment is the first environment that empirically confirms the capabilities of the introduced GA in the ACS.

Butz, Goldberg, and Stolzmann (1999) showed that the ACS performed well in this environment. Unlike XCS, which only forms a complete payoff map of the environment, the ACS is able to build a complete cognitive map of the environment. However, observing the resulting population in the Woods1 environment revealed that the ACS over-specializes its classifiers. Although the classifiers are not completely specialized, there exist attributes that are unnecessarily specialized.

Figure 4 compares the performance of the ACS without and with GA. *Percent Knowledge* refers to the percentage of the so far learned environmental representation. The Population size is the number of different classifiers (i.e. macro-classifiers) in the population. The figure reveals, that the ACS with GA needs slightly more time to generate a complete environmental representation but the gain in generality is enormous. While the ACS without the GA achieves a specificity ratio of 0.41 the ACS with the GA reveals a much better generalization with a ratio of 0.25. Note the change in the specificity when the GA is applied. First, the ALP and the GA explore, resulting in an increase of the specificity. Due to the GA this increase takes longer than in the specificity curve without the GA. When the environmental representation evolves, the GA and the ALP decrease exploration and due to subsumption and deletion the average specificity in the population decreases. Similarly, the population size increases due to the GA at first to a higher level but then decreases and reaches the size of the ACS without the GA. This shows that the ACS with the GA nicely converges to accurate, more general classifiers. A more detailed investigation of the classifier list showed that indeed the accurate, maximally general classifiers are generated and do take over the population.

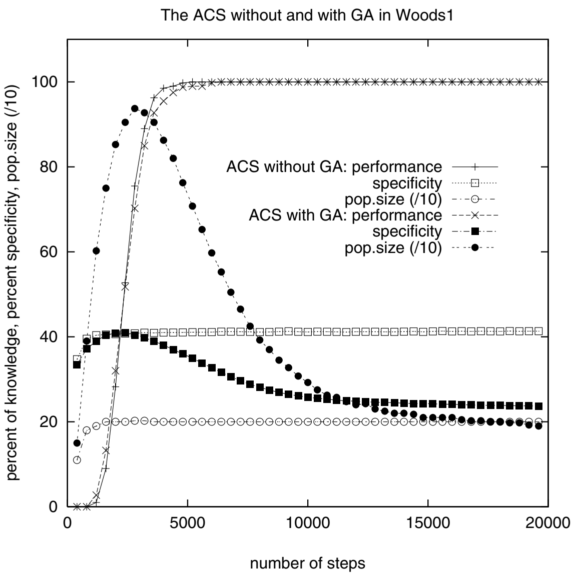

Figure 4: While achieving much more general classifiers, the performance is only slightly decreased when applying the GA. The results are averaged over ten runs.

7 SUMMARY

So far the ACS only learned with the anticipatory learning process (ALP) which represents a specialization pressure. The paper enhanced the application of the ALP to the whole action set. Furthermore, it introduced a first generalization pressure to the ACS mechanism, realized by a genetic algorithm (GA). The interaction between the ALP and the GA was revealed. The results show the possibility of applying GAs in the ACS in order to generate maximally general classifiers without disturbing the ALP. Moreover, it shows the possibility of combining an indirected generalization pressure with a directed specialization pressure in order to generate accurate, maximally general classifiers in LCSs in general. In the Woods1 environment the ACS with genetic generalization pressure generates more general classifiers and the performance is only slightly decreased. In Butz, Goldberg, and Stolzmann (2000) the GA mechanism is further investigated and the abilities of the ACS combined with a GA are revealed.

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 2: Performance Analysis. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. In press.

Butz, M. V., & Stolzmann, W. (1999). Action-planning in Anticipatory Classifier Systems. In *2.International Workshop on Learning Classifier Systems (2.IWLCS) on the Genetic and Evolutionary Computation Conference (GECCO-99)* (pp. 242–249). Orlando, Florida: Morgan Kaufmann.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, *1*, 69–93. (Also TCGA Report 90007).

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

Kovacs, T. (1996). *Evolving Optimal Populations with XCS Classifier Systems*. Master's thesis, School of Computer Science, University of Birmingham, Birmingham, U.K.

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

Stolzmann, W. (1998). Anticipatory Classifier Systems. In *Genetic Programming '98* (pp. 658–664). University of Wisconsin, Madison, Wisconsin: Morgan Kaufmann.

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.