Biasing Exploration in an Anticipatory Learning Classifier Systems

Martin V. Butz

Department of Cognitive Psychology
University of Würzburg
Röntgenring 11
97070 Würzburg, Germany
butz@psychologie.uni-wuerzburg.de

Abstract. The chapter investigates how model and behavioral learning can be improved in an anticipatory learning classifier system by biasing exploration. First, the applied system ACS2 is explained. Next, an overview over the possibilities of applying exploration biases in an anticipatory learning classifier system and specifically ACS2 is provided. In ACS2, a *recency bias* termed *action delay bias* as well as an *error bias* termed *knowledge array bias* is implemented. The system is applied in a dynamic maze task and an hand-eye coordination task to validate the biases. The experiments exhibit that biased exploration enables ACS2 to evolve and adapt its internal environmental model faster. Also adaptive behavior is improved.

1 Introduction

Recently, anticipatory learning classifier systems (ALCS) (Butz, 2001) have gained an increasing interest in the learning classifier system (LCS) community (Lanzi, Stolzmann, & Wilson, 2001). An ALCS is an evolutionary rule learning system that represents anticipations in some form. The consequence is the evolution of an *environmental model* additional or even in contrast to the *payoff model* in other LCSs. The system is able to predict the perceptual consequences of an action in all possible situations in an environment. Thus, the system evolves a model that specifies not only what to do in a given situation but also provides information of what will happen after a specific action was executed.

The major concern in ALCS research during the last few years laid in the optimization of the model learning capabilities. It was intended to evolve a model that is *complete, accurate, and maximally general*. A model that *correctly* represents *all* possible action-effect relations with respect to situational properties with the *least number of maximally general* rules possible. For that, it is necessary to specify which environments are actually solvable with the pursued system. In ALCSs, usually deterministic and discrete Markov environments have been addressed (Gérard & Sigaud, 2001, Stolzmann, 2000). This, however, does not mean that ALCSs are generally limited to such environments as the application to noisy environments (Stolzmann & Butz, 2000), non-Markov environments

(Stolzmann, 2000), and stochastic environments (Butz, Goldberg, & Stolzmann, 2001) has shown.

The exploitation of the evolving knowledge for an improved adaptive behavior has only been pursued to a limited extend so far. Previous model exploitation approaches in ALCSs were published in Stolzmann, Butz, Hoffmann, and Goldberg (2000), enhancing behavioral capabilities by internal reinforcement updates or lookahead action selection, and in Stolzmann and Butz (2000), enhancing model learning capabilities by planning.

In contrast to the global planning approach for improved model learning in Stolzmann and Butz (2000), the approach herein improves model learning by a local, computationally inexpensive method. Biasing exploration towards unknown regions or regions which have not been visited for a long time, model learning as well as adaptive behavior is improved.

The chapter is structured as follows. First, section 2 introduces the applied ALCS system ACS2 and points out the differences to the previous ACS system. Section 3 introduces two exploration bias approaches that are able to increase the model learning capabilities of the system. Section 4 shows that both biases work well in a maze environment. The combination of the two approaches proves to be most robust. Model learning and adaptive behavior is then investigated in an extended dynamic maze. A comparison of ACS2 with the biased exploration to the previous introduced planning approach (Stolzmann & Butz, 2000) in an hand-eye coordination task is presented as well, exhibiting the power of the approach. In this respect, ACS2 is now able to solve larger hand-eye problems much faster. Section 5 summarizes and concludes the chapter.

2 An Introduction to ACS2

Usually, anticipations in an ALCS are represented explicitly in each rule or classifier. That is, each rule consists of a condition-action-effect triple that specifies which perceptual effect takes place after executing the specified action under the specified conditions. Stolzmann (1997) developed a first ALCS (ACS) with an additional anticipatory part in each classifier. Another ALCS system with an explicit anticipatory part, YACS, has been published in Gérard and Sigaud (2001). Tomlinson and Bull (2000) published a cooperate learning classifier system (CXCS) in which cooperations between rules allow anticipatory processes. The ALCS used herein is derived from Stolzmann's work but comprises several important modifications and enhancements. This section introduces the derived system ACS2 and mentions the most important differences to ACS.

Generally, the name ACS2 is meant to introduce a new name for the current state of the ACS system rather than to draw any concrete distinctions. One major difference of ACS2 in comparison with ACS is that ACS2 evolves explicit rules for situation-action tuples in which no change occurs. Moreover, ACS2's anticipatory learning process, which results in a directed specialization pressure, as well as ACS2's genetic generalization mechanism are modified compared to the mechanisms in ACS improving the evolution of accurate, maximally general classifiers.

2.1 Agent architecture

Similar to the agent architectures applied to reinforcement learning approaches or LCSs in particular, ACS2 interacts autonomously with an environment. In a behavioral act at a certain time t, the agent perceives a situation $\sigma(t) \in \mathcal{I} = \{\iota_1, \iota_2, ..., \iota_m\}^L$ where m denotes the number of possible values of each environmental attribute (or feature), $\iota_1, ..., \iota_m$ denote the different possible values of each attribute and L denotes the string length. Note that each attribute is not necessarily coded binary but can only take on discrete values. Moreover, the system can act upon the environment with an action $\alpha(t) \in \mathcal{A} = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ where n specifies the number of different possible actions in the environment and $\alpha_1, ..., \alpha_n$ denote the different possible actions. After the execution of an action, the environment provides a scalar reinforcement value $\rho(t) \in \Re$. Figure 1 illustrates the basic interaction.

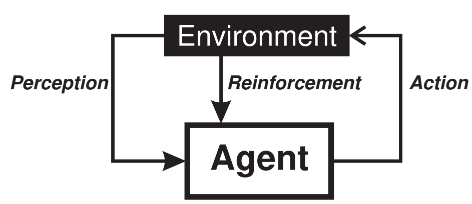

Fig. 1. The basic agent environment interaction

2.2 Knowledge Representation

As in other LCSs, the knowledge in ACS2 is represented by a population $[P]$ of classifiers. Each classifier represents a condition-action-effect rule that anticipates the model state resulting from the execution of the action given the specified condition. A classifier in ACS2 always specifies a complete resulting state. It consists of the following main components.

- *Condition part* (C) specifies the set of situations in which the classifier is applicable.
- *Action part* (A) proposes an available action.
- *Effect part* (E) anticipates the effects of the proposed action in the specified conditions.
- *Quality* (q) measures the accuracy of the anticipated effects.
- *Reward prediction* (r) estimates the reward encountered after the execution of action A in condition C.

– *Immediate reward prediction* (ir) estimates the direct reinforcement encountered after execution of action A in condition C.

The condition and effect part consist of the values perceived from the environment and '#'-symbols (i.e. $C, E \in \{\iota_1, ..., \iota_m, \#\}^L$). A #-symbol in the condition called *don't-care symbol* denotes that the classifier matches any value in this attribute. A '#'-symbol in the effect part, called *pass-through symbol*, specifies that the classifier anticipates that the value of this attribute will not change after the execution of the specified action. Non pass-through symbols in E anticipate the change of the particular attribute to the specified value in contrast to ACS in which a non pass-through symbol did not require a change in value. The action part specifies any action possible in the environment ($A \in \mathcal{A}$). The measures q, r, and ir are scalar values where $q \in [0, 1]$, $r \in \Re$, and $ir \in \Re$. A classifier with a quality q greater than the reliability threshold θ_r (usually set to 0.9) is called *reliable* and becomes part of the internal environmental model. A classifier with a quality q lower than the inadequacy threshold θ_i (usually set to 0.1) is considered as inadequate and is consequently deleted. The immediate reward prediction ir is separated from the usual reward prediction r in order to enable proper internal reinforcement learning updates. All parts are modified according to a reinforcement learning mechanism, and two model learning mechanisms specified in section 2.3.

Additionally, each classifier comprises a *Mark* (M) that records the values of each attribute of all situations in which the classifier did not anticipate correctly sometimes. The *mark* has the structure $M = (m_1, ..., m_L)$. Each attribute $m_i \subseteq \{\iota_1, ..., \iota_m\}$ records all values at position i of perceptual strings in which the specified effect did not take place after execution of action A. Moreover, each classifier specifies a *GA time stamp* t_{ga}, an *ALP time stamp* t_{alp}, an application average aav, an experience counter exp, and a numerosity num. The two time stamps record the time of the last learning module applications. The application average estimates the frequency the classifier is updated (i.e. part of an action set). The experience counter counts the number of applications. The numerosity denotes the number of micro-classifier this *macroclassifier* represents. Thus, one classifier can represent many identical micro-classifiers.

2.3 Environmental Model in ACS2

Interacting with an environment, ACS2 learns more and more about the structure of the environment. Usually, the agent starts without any prior knowledge except for the knowledge implicitly included in the coding structure and the competence in executing the provided actions. Initially, classifiers are mainly generated by a covering mechanism in the anticipatory learning process (ALP). Later, the ALP generates specialized classifiers while a genetic generalization process produces generalized offspring. Moreover, reinforcement learning techniques are applied to the evolving rules representing a behavioral policy in the evolving environmental model.

Figure 2 illustrates the interaction of ACS2 with its environment and its learning application in further detail. After the perception of the current situation $\sigma(t)$, ACS2 forms a match set $[M]$ comprising all classifiers in the population $[P]$ whose conditions are satisfied in $\sigma(t)$. Next, ACS2 chooses an action $\alpha(t)$ according to some action selection strategy. Usually, a simple $\epsilon-$greedy strategy is applied as often used in reinforcement learning (Sutton & Barto, 1998). With respect to the chosen action an action set $[A]$ is generated that consists of all classifiers in $[M]$ that specify the chosen action $\alpha(t)$. After the execution of $\alpha(t)$, classifier parameters are updated by the ALP and the applied RL technique and new classifiers might be generated as well as old classifiers might be deleted by the ALP and genetic generalization. The different learning mechanisms and the various interactions of the mechanisms are explained in the following.

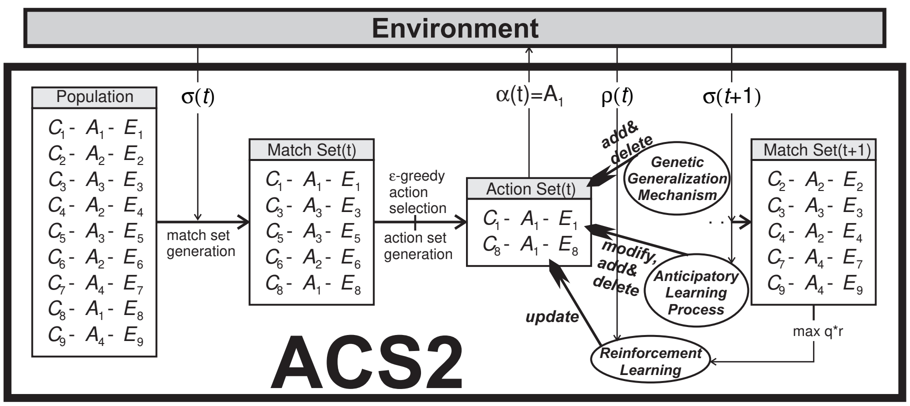

Fig. 2. During one agent/environment interaction, ACS2 forms a match set representing the knowledge about the current perceptions. Next, it generates an action set representing the knowledge about the consequences of the chosen action in the given situation. Classifier parameters are updated by RL and ALP. Moreover, new classifiers might be added and old classifiers might be deleted by genetic generalization and ALP.

Anticipatory Learning Process The ALP was originally derived from the cognitive theory of anticipatory behavioral control (Hoffmann, 1993). The process results in the evaluation and specialization of the anticipatory model in ACS2. Before the generation of offspring, classifier parameters are updated evaluating the anticipations of the classifiers in action set $[A]$. Next, offspring is generated and inaccurate classifiers are deleted.

Parameter Updates The ALP updates the quality q, the mark M, the ALP time stamp t_{alp}, the application average aav, and the experience counter exp. The quality q is update according to the classifiers anticipation. If the effect part of the classifier correctly specified changes and non-changes, the quality is increased by the following formula.

$$q \leftarrow q + \beta(1 - q) \tag{1}$$

On the other hand, if the effect did not specify the correct outcome, the quality is decreased.

$$q \leftarrow q - \beta q \tag{2}$$

In the equation, $\beta \in [0, 1]$ denotes the *learning rate* of ACS2. The smaller the learning rate, the more passive ACS2 updates its values and the less are the values biased towards recent environmental interactions. On the other hand, the larger the learning rate, the faster the parameters adapt to changes in the environment but also the more noisy are the parameters.

Situation $\sigma(t) = (\sigma_1, ..., \sigma_L)$ is added to the mark $M = (m_i, ..., m_L)$ if the classifier did not anticipate correctly. In this case, $\forall_i m_i = m_i \cup \{\sigma_i\}$.

The ALP time stamp is set to the current time t recording the last parameter update in the ALP. Before that, though, the application average *aav* is updated using the *moyenne adaptive modifiée* technique as introduced in Venturini (1994).

$$cl.aav \leftarrow \begin{cases} (cl.aav(cl.exp - 1) + (t - cl.t_{alp}))/cl.exp & \text{if } cl.exp < 1/\beta \\ cl.aav + \beta(t - cl.t_{alp}) & \text{otherwise} \end{cases} \tag{3}$$

The technique assures the fast adaptation of *aav* once the classifier is introduced and later assures a continues update according to the overall learning rate β. Note, this technique also introduces a possible high factor of noise in the young classifier. Thus, the technique is not applied in the quality updates.

Finally, the experience counter *exp* is increased by one.

Classifier Generation and Deletion The ALP generates specialized offspring and/or deletes *inaccurate* classifiers.

Inaccurate classifiers are classifiers whose quality is lower than the inaccuracy threshold θ_i. When the quality of a classifiers falls below θ_i after an update, it is deleted.

More specialized classifiers are generated in two cases. In an *expected case*, in which a classifier anticipated the correct outcome, a classifier might be generated if the mark M differs from the situation $\sigma(t)$ in some attributes, i.e. $\exists_{i,j} \iota_j \in m_i \wedge \iota_j \neq \sigma_i$. Since the mark specifies the characteristics of situations in which a classifier did not work correctly, a difference indicates that the specific position might be important to distinguish the correct and wrong outcome case. Thus, the ALP generates an offspring whose conditions are further specialized. If there are unique differences in the mark compared to the current situation, i.e. $\exists_i \sigma_i \notin m_i$, then one of the unique difference is specialized in the offspring. However, if there are only positions that differ but σ_i is always part of m_i, i.e. $\forall_i \sigma_i \in m_i$, then all differing positions are specialized. The number of specialized positions in the conditions that are not specialized in the effects is limited to u_{max}.

In an *unexpected case*, a classifier did not anticipate the correct outcome. In this case, an offspring classifier is generate, if the effect part of the classifier can be further specialized (by changing pass-through symbols to specific values) to specify the perceived outcome correctly. All positions in condition and effect part are specialized that change from $\sigma(t)$ to $\sigma(t+1)$.

In both reproduction cases, the Mark M of the offspring is emptied, the experience counter *exp* is set to zero, ALP and GA time stamp are set to the current time t, the numerosity is set to one, and all other parameters are inherited from the parents. If the generated offspring already exists in the population $[P]$, the offspring is discarded and the quality q of the old classifier is increased applying equation 1.

A classifier is also generated if there was no classifier in the actual action set $[A]$ that anticipated the effect correctly. In this case, a *covering classifier* is generated that is specialized in all attributes in condition and effect part that changed from $\sigma(t)$ to $\sigma(t+1)$. The covering method was not applied in ACS since in ACS a completely general classifiers was always present for each action. The attributes of the Mark M of the covering classifier are initially empty. Quality q is set to 0.5 as well as the reward prediction r, while the immediate reward prediction ir as well as the application average avv are set to 0. The time stamps are set to the current time t.

Genetic Generalization Mechanism While the ALP specializes classifiers in a quite competent way, over-specializations can occur sometimes as studied in (Butz, 2001). Since the over-specialization cases can be caused by various circumstances, a genetic generalization (GG) mechanism was applied that, interacting with the ALP, results in the evolution of a complete, accurate, and maximally general model. The mechanism works as follows.

After the application of the ALP, it is first determined if the mechanism should be applied. GG is applied if the average time since the last GA application in the current action set $[A]$ is larger than the threshold θ_{ga} (if $t - (\sum_{cl\in[A]} cl.t_{ga} cl.num)/\sum_{cl\in[A]} cl.num > \theta_{ga}$). If the mechanism is applied, θ_{ga} is set to the current time t. Next, two classifiers are selected with roulette wheel selection with respect to the qualities q. The two classifiers are reproduced where the attributes in the mark are emptied, and the qualities are halved. Next, the reproduced classifiers are mutated applying a generalizing mutation, that is, only mutating specified attribute in the condition part C back to don't care symbols. A specialized attribute is generalized with a probability μ. Moreover, conditions of the offspring are crossed applying two-point crossover with a probability of χ. In the case of a crossover application, quality, reward prediction, and immediate reward prediction are averaged over the offspring. Finally, the classifiers are inserted. If a generated offspring already exists in the population, the offspring classifier is discarded and if the existing classifier is not marked its numerosity is increased by one.

The GG mechanism also applies a deletion procedure inside the action sets. If an action set $[A]$ exceeds the action set size threshold θ_{as}, excess classifiers are deleted in $[A]$. The procedure applies a tournament selection process in which the classifier with the significant lowest quality, or the classifier with the highest specificity is deleted. Thus, deletion causes the extinction of low-quality as well as over-specialized classifiers.

Subsumption To further emphasize a proper model convergence, subsumption is applied similar to the subsumption method in XCS (Wilson, 1998). If an offspring classifier was generated, regardless if by ALP or GG, the set is searched for a subsuming classifier. The offspring is subsumed if a classifier exists that is more general in the conditions, specifies the same effect, is *reliable* (its quality is higher than the threshold θ_r), is not marked, and is *experienced* (its experience counter exp is higher than the threshold θ_{exp}). If there are more than one possible subsumer, the syntactically maximally general subsumer is chosen. In the case of a draw, the subsumer is chosen at random from the maximally general ones. If a subsumer was found, the offspring is discarded and either quality or numerosity is increased dependent on if the offspring was generated by ALP or GG, respectively.

Interaction of ALP and GG Several distinct studies in various environments revealed that the interaction of ALP and GG is able to evolve a complete, accurate, and maximally general model in various environments in a competent way (see e.g. Butz, Goldberg, & Stolzmann, 2000, Butz, 2001). The basic idea behind the interacting model learning processes is that the specialization process extracts as much information as possible from the encountered environment continuously specializing over-general classifiers. The GG mechanism, on the other hand, randomly generalizes exploiting the power of a genetic algorithm where no more additional information is available from the environment. The ALP ensures diversity and prevents the loss of information of a particular niche in the environment. Only GG generates identical classifiers and causes convergence in the population.

2.4 Policy in ACS2

The behavioral policy of ACS2 is directly represented in the evolving model. As specified above, each classifier includes a reward prediction estimate r and an immediate reward prediction estimate ir. Thus, as explained in Butz (2001), the environmental model needs to be specific enough to be able to represent a proper policy in the model. If this is not the case, *model aliasing* might take place.

In model aliasing a model is completely accurate in terms of anticipations but over-general with respect to reinforcement. For example, in a game such as checkers, ACS2 might be able to learn when it is possible to move a figure in some direction. However, the model will not be specific enough to evolve a proper strategy directly represented in the model. In checkers, it is sufficient to specify that the targeted position is currently empty in order to know the success of a simple movement. However, to be able to determine if the movement will payoff, more positions of the own as well as the opponents pieces need to be specified.

Policy Learning After the action was executed, the next environmental situation was perceived and the subsequent match set was formed as visualized in figure 2, the reward related parameters r and ir are updated.

$$r \leftarrow r + \beta(\rho(t) + \gamma \max_{cl \in [M](t+1) \wedge cl.E \neq \{\#\}^L} (cl.q \cdot cl.r) - r) \tag{4}$$

$$ir \leftarrow ir + \beta(\rho(t) - ir) \tag{5}$$

As before, $\beta \in [0, 1]$ denotes the learning rate biasing the parameters more or less towards recently encountered reward. $\gamma \in [0, 1)$ denotes the discount factor similar to Q-learning (Watkins, 1989). In contrast to Q-learning, however, the rules may be applicable in distinct situations. The values of r and ir consequently specify an average of the resulting reward after the execution of action A over all possible situations of the environment in which the classifier is applicable.

Policy Execution Usually, ACS2 applies a simple ϵ-greedy action selection strategy. An action is chosen at random with a probability ϵ and otherwise the best action is chosen. The action with the highest $q * r$ value in a match set $[M]$ is usually considered as the best action in ACS2.

While the reinforcement learning capabilities of ACS2 will be of minor interest in the following sections, the modification of the policy will be of major interest. The question is how the model learning progress can be further optimized by modifying the behavioral policy.

3 Model Exploitation

While diverse experiments have shown ACS2's capabilites of evolving a complete, accurate, and maximally general environmental model reliably (Butz, Goldberg, & Stolzmann, 2000), the approach herein optimizes the capability further. It is investigated, how the policy can be optimized to improve sampling of the environment and consequently optimize model learning in return.

The model building algorithm in ACS2 is basically an *implicitly supervised* learning algorithm. This means that the perceptual feedback of the environment can be regarded as a supervision but no explicit teacher is necessary for the supervision. In fact, the environment implicitly represents the supervisor. The agent does not ask for corrections but extracts information from the environment actively. Moreover, it manipulates the perception of environmental feedback by its own actions. Thus, an action policy that optimizes feedback can improve the model learning capabilities of ACS2. The optimization of the information extraction process by the means of an intelligent action selection process is the concern of the model exploitation approach herein.

3.1 Background

Previous intelligent situation and action selection approaches have been pursued in the RL literature. Sutton (1990) published the idea of an exploration bonus in his dynamical architecture *Dyna* that allowed the faster detection of environmental dynamics and a consequent faster adaptive behavior. In his approach, the agent is *positive* in that it believes that it is worth to explore unknown regions of its environment. In more behavioral terms, the agent might be said to be curious and feel save. Moore and Atkeson (1993) introduced a prioritized sweeping algorithm that further improved the internal update strategy in Dyna. Their approach maintains a priority queue of to-be-updated states with the order of updating states in which previously large changes in the absorbing state prediction or the reward prediction values occurred. Dayan and Sejnowski (1996) systemize and extend Sutton's (1990) exploration bonus approach. By modeling the uncertainty of the world and providing the agent with additional goal information, their enhanced Dyna system is able to directly explore the existence of barriers and consequently detect possible shortcuts faster. Finally, Kaelbling (1993) applied an interval estimation strategy in which actions are selected according to prediction and the estimated error of the prediction.

Thrun (1992) as well as Wilson (1996) provide excellent overviews of the underlying exploration/exploitation dilemma and the different types of possible exploration. According to their frameworks, the model exploitation approach herein is a *directed* exploration approach pursuing *recency-based methods* as well as *error-based* methods. Directed exploration is compared to *undirected uniform random* exploration. The usual trade-off between exploration and exploitation does not apply in most of the experiments herein. In the trade-off problem, exploration usually inhibits the proper exploitation of the current knowledge for an optimal behavior. This work focuses on model learning optimization rather than policy learning optimization so that optimal behavior is purely defined as the behavior of optimizing knowledge, and not behavior, as fast as possible. However, the results in section 4.3 show that the direct bias is also able to positively influence policy learning in which behavior is realized by a simple *constant global* explore/exploit strategy.

3.2 Approach

The idea of the biased exploration is to decide before the execution of an action, from which action the system probably learns the most. To do that, the classifiers in the current match set $[M]$ are searched for indications which action might result in the highest knowledge gain. The action that promises the highest knowledge gain is then executed during exploration rather than a randomly chosen action.

Action Delay Bias The first bias exploits the recency-based bias principle. The execution of actions which have been executed quite long ago in a given situation promises the detection of new situation-action-effect relations. This is similar to the exploration bonus approach pursued in Sutton (1990). In ACS2, such a bias needs to be adapted to the generalized evolving model representation.

To determine which action has been executed most long ago in ACS2, the mechanism takes advantage of the existing ALP time stamp t_{alp}. Given a current situation $\sigma(t)$ and the corresponding match set $[M](t)$, the action of the classifier with the lowest value t_{alp} in $[M]$ is assumed to denote the action that was chosen least recently in $\sigma(t)$. However, if there is an action that is not represented by any classifier in $[M]$, this action is assumed to be the one that was experienced most long ago (or not at all, yet) so that this action is chosen for execution.

Knowledge Array Bias The second exploration bias is based on the error-based bias principle. Moore and Atkeson (1993) implemented this bias with respect to absorbing state prediction error and reward prediction error. Our bias however is concerned with optimizing model learning and consequently considers the anticipatory error.

The anticipatory error is actually expressed in the quality q of each individual classifier since the quality denotes the accuracy of the specified anticipations. Thus, a knowledge array bias application directly infers from the classifiers in the match set $[M]$, which action result is known the least. The bias uses the knowledge represented in all classifiers. Similarly to the prediction array PA in the XCS classifier system (Wilson, 1995) a knowledge array KA is generated in ACS2. KA has an entry for each possible action a that specifies the knowledge about the consequences of the action. It is determined as follows:

$$KA[a] = \sum_{cl \in [M] \wedge cl.A = a} cl.q \cdot cl.num / \sum_{cl \in [M] \wedge cl.A = a} cl.num \tag{6}$$

Thus, each entry specifies the averaged quality of the anticipation for each possible action. The action with the lowest value in the knowledge array is chosen for execution. As in the action delay bias, actions that are not represented by any classifier are always chosen for execution first.

Interacting Biases During exploration, an action is chosen according to the applied bias with a fixed biased exploration probability p_b while otherwise an action is chosen as before uniformly randomly. Section 4 validates the approach in a dynamic maze environment as well as an hand-eye coordination task.

Since the two approaches actually comprise different biases with the same objective, that is the optimization of model learning, action delay bias and knowledge array bias are also applied in combination. In this case, with a probability of p_b, action delay bias is executed with a probability of 0.5 and knowledge array bias is executed otherwise. Section 4.1 actually proves that the combined bias generates the best learning results as well as remedies possible misguiding effects.

4 Experimental Validation

The different exploration biases are now evaluated in a maze learning task as well as a hand-eye coordination task. Maze tasks have been studied widely in the LCS literature (see e.g. Lanzi, 1999) and RL literature (see e.g. Moore & Atkeson, 1993). The hand-eye coordination task was introduced in Birk (1995) and was previously studied with ACS in Stolzmann and Butz (2000). The accompanying results show that by biasing exploration ACS is able to achieve a complete, accurate, and compact knowledge of the environment faster. Moreover, it is shown that behavior is adapted faster. All model learning curves are averaged over twenty runs. [1]

4.1 Exploration Bias in Maze5

The Maze5 environment has been studied before in Lanzi (1999) with XCS investigating the reinforcement learning capabilities of the system. The left hand side of figure 3 depicts Maze5. The environment is a discrete, deterministic environment. The agent perceives the eight adjacent cells so that $\sigma(t) \in \mathcal{I} = \{., O, F\}^8$. Moreover, the agent is able to move to those eight adjacent cells so that $\alpha(t) \in \mathcal{A} = \{N, NE, E, SE, S, SW, W, NW\}$. A movement towards an obstacle O has no perceptual effects and the animat remains in its current position. Movement into the food position F results in the provision of a reinforcement of 1000, the perception in the food position, and finally an end of trial and a consequent reset of the agent to a randomly chosen empty position. The experimental results in this and the following section display the model learning progress of the system. To evaluate the progress, all *reliable* classifiers of the current population are searched for a classifier that anticipates the correct changes in each situation-action tuple that actually does induce a change in the environment. Situation-action tuples that do not cause a change are not tested since the eight classifiers for these cases are learned very quickly. If those cases were included, the result would be an initial steep increase in knowledge in the beginning of a learning curve, which would only complicate the monitoring of the knowledge progress in the changing cases.

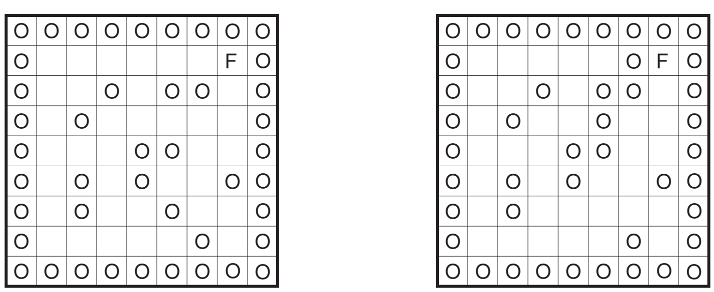

Fig. 3. Maze5 on the left-hand side and Maze7 on the right-hand side. Both mazes are discrete, deterministic Markov environments.

[1] If not states differently, parameters were set to: $\beta = 0.05$, $u_{max} = \infty$, $\gamma = 0.95$, $\theta_{ga} = 100$, $\mu = 0.3$, $\chi = 0.8$, $\theta_{as} = 20$, $\theta_{exp} = 20$, $\epsilon = 1.0$.

Figure 4 shows that action delay bias, knowledge array bias, as well as the combination of the biases positively influence the shape of the learning curves. Knowledge array bias shows a stronger effect than action delay bias. However, in both cases the application of a completely biased exploration ($p_b = 1.0$) disrupts the learning progress. In this case, a misguiding effect was observed that caused the system to neglect parts of the environment. Consequently, a complete knowledge of the environment was not achievable. The combination of the two biases however shows the best learning effect. It reliably generates a complete knowledge even with a $p_b = 1.0$. The remainder of this work consequently only shows results with the combined biases with $p_b = 1.0$.

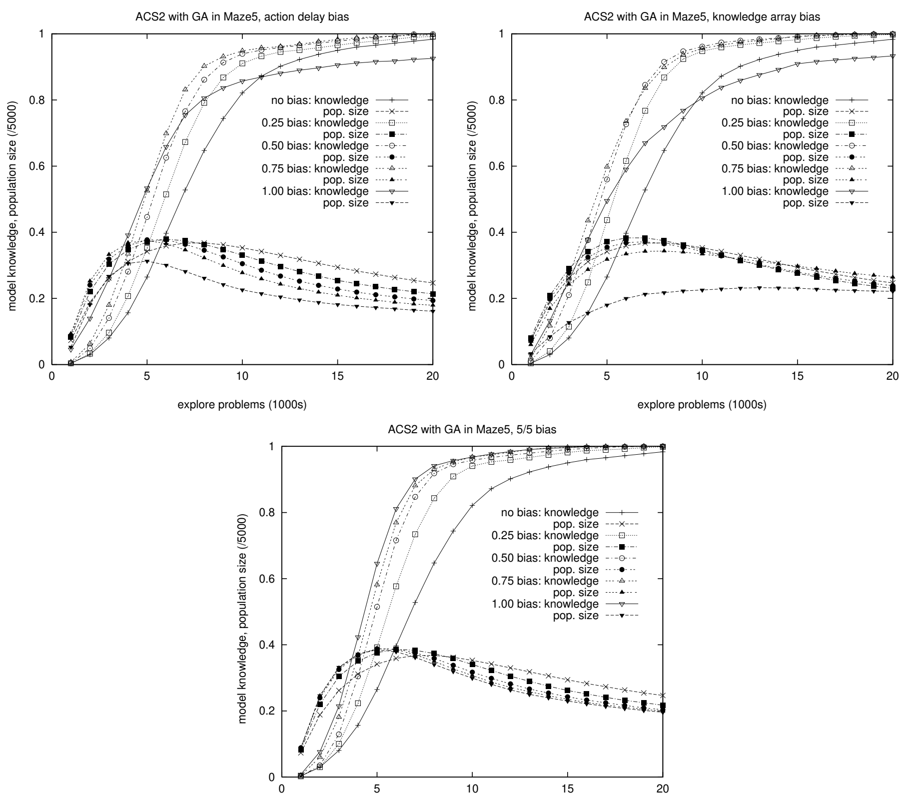

Fig. 4. Each bias causes a positive learning effect. Combining action delay bias and knowledge array bias, the best learning progress as well as the highest robustness is achieved.

4.2 Model Learning in a Dynamic Maze

The next step in the investigation of the additional exploration bias is to change Maze5 to a dynamic maze. In figure 3, Maze7 is visualized on the right-hand side. Maze7 is more difficult in a reinforcement learning task, because longer chaining is necessary. The dynamics are simulated by switching back and forth between Maze5 and Maze7 consequently always altering the maze structure slightly. The presented model learning experiments always start in the Maze5 environment and switch every 20000 steps to Maze7 and back to Maze5.

Figure 5 shows that the additional exploration bias always increases the model learning speed and the consequent knowledge adaptation in the dynamic environment. It is also observable that ACS2 is able to remember parts of the other environment in its classifier population since the knowledge loss is smaller and the recovery speed is faster in the later switches between Maze5 and Maze7. Note also that the peaks as well as the minima in the population size increasingly decrease indicating that the internal representation becomes increasingly compact.

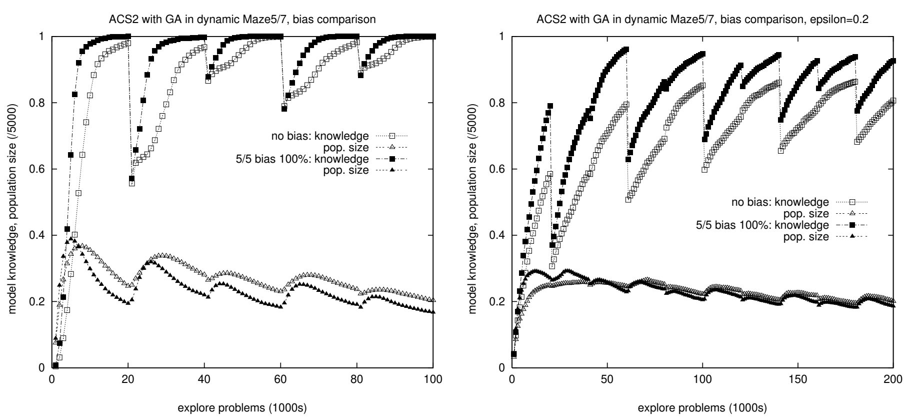

Fig. 5. In the dynamic Maze5/7 environment, ACS2 is able to adapt its knowledge faster with additional exploration bias. Although the knowledge is more complete, the population size is smaller than in the case where random action choice is applied. Decreasing the exploration probability ϵ, the model learning speed decreases in general.

On the right hand side the model learning behavior of ACS2 is observable when ϵ is decreased. As would be expected, ACS2 builds up its environmental knowledge much slower in this case. The positive effect of exploration bias instead of random exploration is still observable, though. Note that in all curves the population size actually decreases more when the exploration bias is applied although a higher knowledge is achieved. The next section investigates the behavioral adaptation of ACS2 in Maze5/7 in which ϵ is altered as well.

4.3 Adaptive Behavior in the dynamic Maze5/7

The behavior of ACS2 is directly represented in its classifiers. The reward prediction of each classifier combined with the quality determines the action choice during exploitation. Thus, it is clear that a proper model representation needs to be present, before an appropriate policy can be represented in the model. However, the model specifies all situation-action tuples whereas for a proper policy it might be sufficient to only represent the important situation-action tuples. The following results show how ACS2 is able to adapt its policy in the dynamic Maze5/7 environment. The first experiment was done according to the usually applied schema in LCSs with one pure exploration and one pure exploitation trial. During pure exploration, ϵ is set to one allowing for the strongest model and consequent policy learning. In the second experiment only one phase is executed during which ϵ is set to 0.2. The steps to food during these ϵ-greedy trials are recorded. Due to the slightly higher noise in the adaptive behavior tasks, the curves are averaged over 50 runs.

The left-hand side of figure 6 shows that ACS2 in the two phase experiment is able to adapt its behavior perfectly in the dynamic Maze5/7 environment. Especially in the switch from Maze5 to Maze7 a faster adaptation is observable when exploration bias is applied. The shortcut detection when switching back to Maze5 is detected nearly equally fast. However, in the ϵ-greedy experiment displayed on the right-hand side of figure 6, ACS2 is actually detecting the short cuts faster when biased exploration is applied. Improved adaptive behavior is observable in general. The curves also reveal that although the model might not yet be complete, behavior is already adapting indicating the simultaneous evolution of model and behavior. Note, the curves also expose that behavior is adapted faster in later environmental changes confirming the retention of previous knowledge.

4.4 Improved Performance in an Hand-Eye Coordination Task

To validate the general benefit of exploration bias, ACS2 was also applied in an hand-eye coordination task previously investigate in Stolzmann and Butz (2000). Figure 7 shows the problem on a 3×3 matrix. A camera is watching a plain on which one block is situated. A gripper acts upon the plain. The gripper is able to move to its four adjacent positions as well as execute a gripping and releasing action ($\mathcal{A} = \{N, E, S, W, G, R\}$). Perceived is the camera image which is discretized into the actual matrix size as well as an additional feeling sensor which indicates if there is currently no contact to the block (indicated by $'0'$), if the block is situated under the gripper (indicated by $'1'$), or if the gripper holds the block ($'2'$). Considering a problem with a matrix size of $l \times l$, the generally possible perceptions are in $\mathcal{I} = \{w, b, g\}^{l^2} \cdot \{0, 1, 2\}$.

Tested in this environment is the environmental model. Thus, as before, only the reliable classifiers are tested. However, not all possible situation-action combinations are tested but randomly generated ones. In a test, 100 randomly chosen situations are generated with a 50% chance that the gripper is currently holding

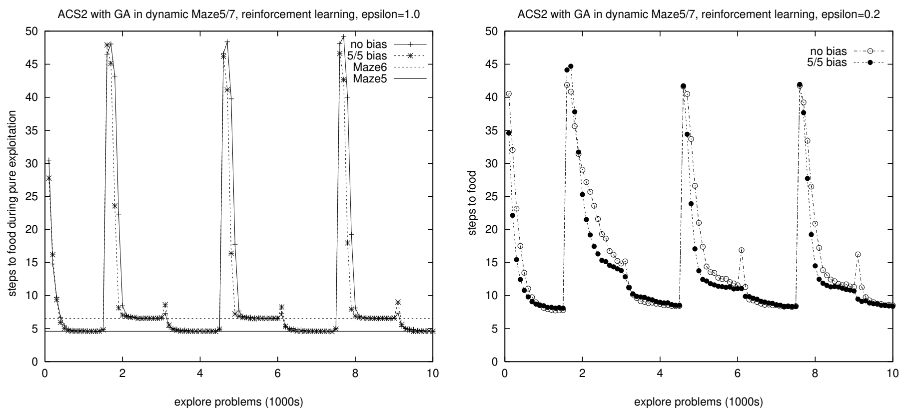

Fig. 6. In the reinforcement learning task in the dynamic Maze5/7 environment, ACS2 is also able to adapt its behavior faster with additional exploration bias. Applying an ϵ-greedy action selection policy, exploration bias enables a faster short cut exploitation and consequently adapts faster in the change from Maze7 to Maze5.

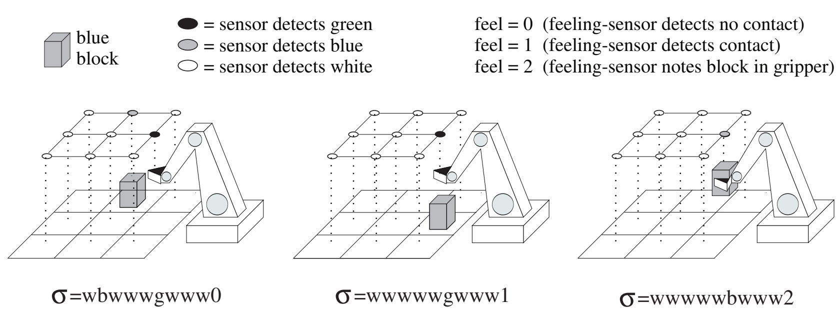

Fig. 7. The hand-eye coordination task exemplified in a 3x3 matrix

the block. Next, an action is chosen at random and the effect is generated. The resulting situation-action-effect triple is tested regardless if a change is caused by the chosen action or not. Note that in this problem the model size grows in $O(l^2)$ whereas the problem size grows in $O(l^4)$. Parameters were set as before and the curves are again averaged over twenty runs.

In Stolzmann and Butz (2000) planning was applied to improve model learning of ACS in the hand-eye scenario. In the planning approach, every 50 steps a goal is requested from a goal generator, a sequence is planned to the goal by a bidirectional search mechanism and if a sequence was found it is executed. Planning is executed as long as sequences are found and are successfully executed. The goal generator as well as the whole planning approach are implemented as specified in (Stolzmann & Butz, 2000).

The left hand side of figure 8 shows performance of ACS2 in the hand-eye task with $l = 5$ comparing the pursued additional exploration bias approach with the previous planning approach, and random exploration. As in the previous work, the curves confirm that the high-level planning approach is able to improve performance compared to simple continuous random exploration. However, planning is an expensive procedure and the effect is rather small. The exploration bias approach on the other hand is able to significantly increase the learning progress compared to both random exploration and the planning approach. Thus, in this case a low-level intelligent action selection strategy was able to beat a high-level computationally expensive planning strategy.

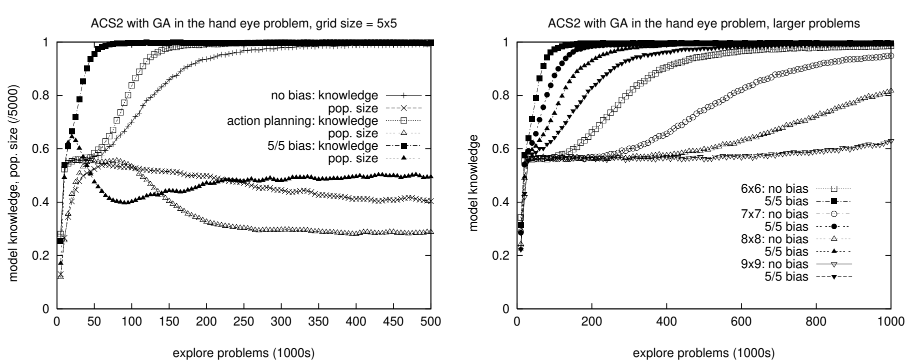

Fig. 8. The local direct exploration bias shows a stronger benefit than the planning strategy which again exhibits faster learning than simple random exploration. In larger problems displayed on the right hand side, ACS2 with exploration bias proves to be able to evolve its knowledge much faster.

The right hand side of figure 8 exhibits performance of ACS2 in larger hand-eye problems. It can be observed that ACS2 strongly benefits from the exploration bias approach. Essentially, exploration bias helps to explore the cases *with contact to the gripper* and *holding the gripper* which occur only rarely when choosing actions at random. This helps ACS2 to learn the cases *transporting* and *releasing the gripper* which are tested during the test phase with a probability of 50%. Although also the planning approach was supposed to lead to the exploration of these cases, exploration bias works more effective and more reliable. A generalized environmental model is evolved much faster than with previous techniques.

5 Summary and Conclusions

Action delay bias as well as knowledge array bias demonstrated to be able to speed up the model learning process in ACS2. The combination of both exploration biases led to the fastest and most stable performance gain. The combination of the biases also showed the highest robustness achieving a complete knowledge with any parameter setting. The application of the combined bias to the dynamic maze environment exhibited further improvement. The dynamic maze task also confirmed that ACS2 is able to retain information as well as reuse information. Also adaptive behavior was improved due to exploration bias. In the dynamic maze task, ACS2 adapted its behavior faster regardless if testing behavior in a separate pure exploitation phase or online applying an ϵ-greedy action selection policy. The application in the hand-eye coordination task proved the robustness of the exploration bias as well as its broad applicability. Compared to the previous high-level planning approach, ACS2 appeared to improve performance significantly. In larger hand-eye problems, ACS2 with exploration bias showed an even stronger improvement in the learning rate.

The exploration bias approach has the advantage of being local and computationally rather inexpensive in contrast to the previous planning approach. In behavioral terms, the applied biases can be compared with *curiosity*, the drive to do things the consequences of which are not known for sure. While the biases were applied in an ALCS, the approach is not restricted to ALCSs and should also result in a significant learning benefit in other LCSs.

Although ACS and in particular now ACS2 demonstrated to learn a complete, accurate, and compact environmental model of diverse problems competently and reliably, what remains to be shown is how to translate this knowledge into behavioral competence. Hoffmann (1993) and other researchers in cognitive sciences emphasize more and more the importance of anticipations for the acquisition of a competent behavior. Right now, however, ACS2 is still rather stimulus-response driven in accordance with behaviorism and reinforcement learning. The applied exploration bias might be regarded as an anticipatory behavior—the anticipation of the (most unknown) consequences influences behavior and might be compared to curiosity, also an anticipatory driven behavior. However, other behaviors should be investigated such as further anticipatory driven action selection processes or the simulation of attentional processes. Future research will show in what way anticipatory learning mechanism like ACS2 can be exploited for the realization of anticipatory behavior in artificial agents.

Acknowledgments I would like to thank the Department of Cognitive Psychology at the University of Würzburg for their ideas and thoughts provided with respect to this work. Moreover, I would like to thank Stewart W. Wilson for pointing out relevant literature and providing many useful comments. The work was sponsored by the German Research Foundation DFG.

References

Birk, A. (1995). *Stimulus Response Lernen [Stimulus response learning].* Doctoral dissertation, University of Saarbrücken, Germany.

Butz, M. V. (2001). *Anticipatory learning classifier systems.* Genetic Algorithms and Evolutionary Computation. Boston, MA: Kluwer Academic Publishers.

Butz, M. V., Goldberg, D. E., & Stolzmann, W. (2000). Introducing a genetic generalization pressure to the anticipatory classifier system: Part 2 - performance analysis. In Whitely, D., Goldberg, D. E., Cantu-Paz, E., Spector, L., Parmee, I., & Beyer, H.-G. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)* pp. 42–49. San Francisco, CA: Morgan Kaufmann.

Butz, M. V., Goldberg, D. E., & Stolzmann, W. (2001). Probability-enhanced predictions in the anticipatory classifier system. In Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.), *Advances in Learning Classifier Systems, LNAI 1996* pp. 37–51. Berlin Heidelberg: Springer-Verlag.

Dayan, P., & Sejnowski, T. J. (1996). Exploration bonus and dual control. *Machine Learning*, *25*(1), 5–22.

Gérard, P., & Sigaud, O. (2001). YACS: Combining dynamic programming with generalization in classifier systems. In Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.), *Advances in Learning Classifier Systems, LNAI 1996* pp. 52–69. Berlin Heidelberg: Springer-Verlag.

Hoffmann, J. (1993). *Vorhersage und Erkenntnis: Die Funktion von Antizipationen in der menschlichen Verhaltenssteuerung und Wahrnehmung. [Anticipation and cognition: The function of anticipations in human behavioral control and perception.].* Goettingen, Germany: Hogrefe.

Kaelbling, L. P. (1993). *Learning in embedded systems.* Cambridge, MA: MIT Press.

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

Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.) (2001). *Advances in learning classifier systems, LNAI 1996.* Berlin Heidelberg: Springer-Verlag.

Moore, A. W., & Atkeson, C. (1993). Memory-based reinforcement learning: Converging with less data and less real time. *Machine Learning*, *13*, 103–130.

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

Stolzmann, W. (2000). An introduction to anticipatory classifier systems. In Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.), *Learning Classifier Systems: From Foundations to Applications, LNAI 1813* pp. 175–194. Berlin Heidelberg: Springer-Verlag.

Stolzmann, W., & Butz, M. V. (2000). Latent learning and action-planning in robots with anticipatory classifier systems. In Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.), *Learning Classifier Systems: From Foundations to Applications, LNAI 1813* pp. 301–317. Berlin Heidelberg: Springer-Verlag.

Stolzmann, W., Butz, M. V., Hoffmann, J., & Goldberg, D. E. (2000). First cognitive capabilities in the anticipatory classifier system. In Meyer, J.-A., Berthoz, A., Floreano, D., Roitblat, H., & Wilson, S. W. (Eds.), *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior* pp. 287–296. Cambridge, MA: MIT Press.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning* pp. 216–224. San Mateo, CA: Morgan Kaufmann.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction.* Cambridge, MA: MIT Press.

Thrun, S. B. (1992). The role of exploration in learning control. In White, D.A. adn Sofge, D. (Ed.), *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches* New York, NY: Van Nostrand Reinhold.

Tomlinson, A., & Bull, L. (2000). A corporate XCS. In Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.), *Learning Classifier Systems: From Foundations to Applications, LNAI 1813* pp. 195–208. Berlin Heidelberg: Springer-Verlag.

Venturini, G. (1994). Adaptation in dynamic environments through a minimal probability of exploration. 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. 371–381. Cambridge, MA: MIT Press.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards.* Doctoral dissertation, King's College, Cambridge, UK.

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

Wilson, S. W. (1996). Explore/exploit strategies in autonomy. In Maes, P., Matariac, M. adn Pollak, J., Meyer, J.-A., & Wilson, S. (Eds.), *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* pp. 325–332. Cambridge, MA: MIT Press.

Wilson, S. W. (1998). Generalization in the XCS classifier system. In Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D., Grazon, M., Goldberg, D., Iba, H., & Riolo, R. (Eds.), *Genetic Programming 1998:*

Proceedings of the Third Annual Conference pp. 665–674. San Francisco: Morgan Kaufmann.