Action-Planning in Anticipatory Classifier Systems

Martin Butz
Institute for Psychology III
University of Wuerzburg
Roentgenring 11
97070 Wuerzburg, Germany

Wolfgang Stolzmann
Institute for Psychology III
University of Wuerzburg
Roentgenring 11
97070 Wuerzburg, Germany

Abstract

Learning consists in the acquisition of knowledge. In Reinforcement Learning this is knowledge about how to reach a maximum of environmental reward. We are interested in the acquisition of knowledge that consists in having expectations of behavioral consequences. Behavioral consequences depend on the current situation, so it is necessary to learn in which situation S which behavior/reaction R leads to which behavioral consequences C. In other words, SRC units are learned.

It was the psychologist Edward Tolman (1932) who firstly stated that animals learn SRC units. Seward (1949) proved that rats are able to learn in the absence of reward and confirmed Tolman’s assumption. Learning in the absence of reinforcement is called ‘latent learning’ and cannot be explained by usual reinforcement learning techniques. In the field of Learning Classifier Systems (LCS) latent learning is realized in Riolo’s CFSC2 (Riolo, 1991) and Stolzmann’s ACS (Stolzmann, 1997, 1998). Both authors prove the performance of their learning algorithms with a simulation of Seward’s experiment. This experiment consists in a learning phase without any reward followed by a test phase where the rats have to use the knowledge they acquired during the learning phase to do action-planning. Action-planning and latent learning occur at different times.

This paper focuses on the integration of action-planning and latent learning in ACS. Using an example about learning of the hand-eye co-ordination of a robot arm in conjunction with a camera it will be shown, that a combination of action-planning and latent learning in ACS induces a substantial reduction of the number of trials which are required to learn a complete model of a prototypically environment.

1 INTRODUCTION

On the basis of learning experiments with rats Edward Tolman (1932) formed a learning theory in which the formation of situation-reaction-consequence units (SRC units), called *field expectancies*, was a basic assumption. These units represent the knowledge of an organism that in situation S the action R leads to the consequences C. Seward (1949) empirically confirmed this assumption and further showed the ability of latent learning with such SRC units. Latent learning is defined as learning without getting any reward. Therefore it cannot be simulated with normal reinforcement learning techniques (cf. Sutton & Barto 1998). In Seward's experiments rats were put into a simple T-maze with distinguishable end-boxes during a latent learning phase and explored the maze without getting any reward. After the learning phase they were fed in one of the end-boxes for one minute and then were placed in the starting point. 28 of 32 rats found the correct path from this point to the desired end-box. This result shows, that already during the latent learning phase the rats must have learned an internal world model.

For being able to build such world models and to do latent learning in Learning Classifier Systems (LCS) analogous to those findings a representation of behavioral consequences is necessary. Such a representation was firstly introduced by Riolo in his CFSC2 (1991). In Anticipatory Classifier Systems (ACS) a C^{ant}-part exists, that explicitly represents anticipated behavioral consequences (Stolzmann 1997,1998). Apart from the ability of latent learning of world models, the possibility of building plans due to the monitored consequences is discussed in CFSC2 and ACS. Though, both approaches used planning only at the end of the latent learning phase to test the achieved knowledge. A comparison of ACS and CFSC2 is given by Stolzmann (1998).

This paper focuses on the integration of action-planning and latent learning in ACS. It is merging the planning possibility with the latent learning process in ACS, building an action-planning phase that will alternate with the normal exploration phase.

By now ACS were used to solve the above mentioned experiment of Seward (1949). Besides it was tested on learning the hand-eye coordination of a robot in conjunction with a camera introduced by Birk (1995). A camera is monitoring a matrix representing a planar surface with a robot-arm on it. The robot is able to simply move in the four directions of the planar surface, grip a block, transport it to a different position and release it again. It has to learn the effects of its possible motor actions.

This scenario is especially useful to show the positive effects of action-planning, because the different states do not occur with the same probability. Only on the position where the block is situated gripping will be successful, so that the system normally will move around the field without the block most of the time. Latent learning of how to grip the block and to move it to a different position seems to be impossible. Besides, it will be possible to compare the positive effects of action-planning in this paper to those Birk made in his work. Finally Birk tested this experiment also on a real robot arm in combination with a real camera, which shows the possible application in reality.

In the next section first of all a short introduction to ACS is given. After that the evaluated experiment is introduced. The fourth section is about the action-planning method added to ACS followed by the experimental results in section 5 and the conclusions.

2 INTRODUCTION TO ANTICIPATORY CLASSIFIER SYSTEMS (ACS)

ACS are a new kind of Learning Classifier Systems (LCS) introduced by Stolzmann (1997). They learn by using the cognitive mechanism of anticipatory behavioral control, as shown in figure 1. It is based on the theories of a learning mechanism by Hoffmann that was firstly introduced in cognitive psychology (1992). For this additional mechanism ACS is able to learn latently, apart form the common reward learning.

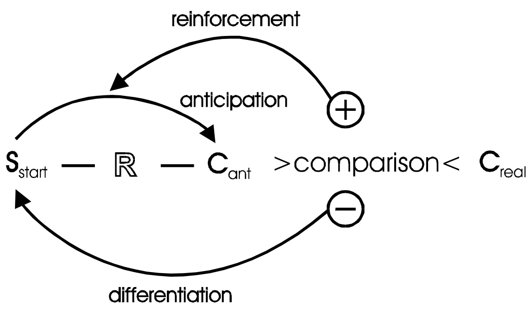

Figure 1: Hypothetical learning mechanism of anticipations for behavioral control (Hoffmann, 1993, p. 44)

2.1 BASIC ELEMENTS

ACS consist of four basic components:

- An *input interface* with detectors for sensory input from the environment,
- an *output interface* for motor actions,
- a *message list* which contains the messages sent by the detectors and the classifier list and
- a *classifier list* which consists of simple production rules called classifiers.

The difference to normal classifiers is that the classifiers in ACS consist of three parts:

- a *condition-part S* with conditions of messages, i.e. with attributes of environmental states that define a situation;
- a *reaction-part R* with instructions for the effectors, i.e. with instructions for motor actions,
- a *consequence-part* C^{ant} with attributes of the anticipated new environmental state, i.e. with attributes of the anticipated behavioral consequences,

and a *mark* that lists all different values of each component of the states in that the classifier did not work correctly.

Therefore a classifier is represented as a 3-tuple (S,R,C^{ant}), with $S, C^{ant} \in \{s_1,...,s_m, \#\}^L$ and $R \in \{r_1,...,r_n\}$, where L is the number of detectors , m is the number of the possible values of each detector and n is the number of different behavioral actions. A #-symbol at the ith component of *S* means that the classifier does not care about the information of the ith detector. A #-symbol at the ith component of C^{ant} means that the classifier system believes that its action *R* does not change the environmental attribute that is perceived by the ith detector. *S*-parts are equal to condition-parts of other LCS. C^{ant}-parts are similar to action-parts of other LCS that use the passthrough process that was defined by Forrest (1985). The only difference is, that C^{ant}-parts may not contain information for the output-interface.

2.2 BEHAVIORAL ACTS IN ACS

The learning mechanism of ACS works as follows (see also Fig. 2):

1. The detectors send their current perceptions to the message list, so that the message list contains an internal representation of the current environmental state.
2. This message m_1^{real} is compared with the *S*-part of all classifiers of the classifier list. Classifiers that match with m_1^{real} are called satisfied classifiers.
3. The ACS decides which satisfied classifier c_i will become active either by running a strength based competition or by simply selecting c_i randomly.
4. The selected classifier c_i becomes active. With regard to m_1^{real} and its C^{ant} -part c_i adds a new message m_2^{ant} to the message list. So m_2^{ant} is the anticipated environmental state.

5. c_i sends its *R*-part r_i to the output-interface which causes a motor action.
6. The detectors send the new internal representation of the environmental state m_2^{real} to the message list.
7. The system compares m_2^{real} with m_2^{ant}.
8. Depending to the result of this comparison the classifier list is modified.
9. m_2^{real} is the new m_1^{real} : to2.

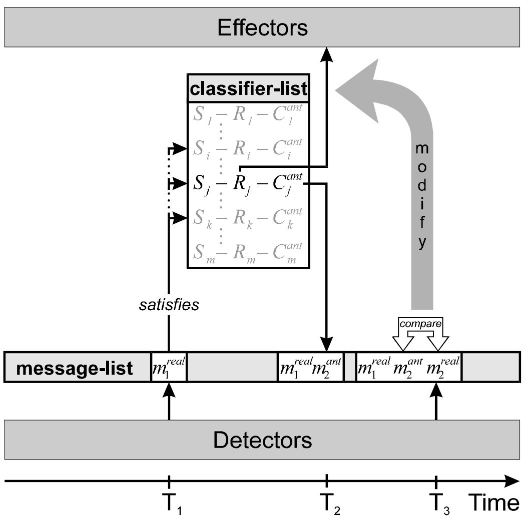

Figure 2: A behavioral act in an ACS (Stolzmann, 1997, p. 70)

2.3 LEARNING IN ACS:

Learning is realized in point 8. The following mechanisms are possible:

1. The classifier did anticipate correctly (expected case):
   - The strength of the classifier is increased.
   - If there is a mark, then a new classifier is formed that is more specific in the *S*- and C^{ant}-parts, trying to exclude the possibility to be chosen in the state(s) described by the mark.
2. The classifier did not anticipate correctly (case of correctable assumptions and expectations / useless case):
   - The classifier gets a mark.
   - The strength of the classifier is decreased.
   - If possible, a new classifier is generated that is more specific than the old one and that anticipates the environment correctly.

As an ACS always starts with completely general rules and specializes these rules step by step no generalizing process is necessary.

2.4 SPECIAL NOTES

Especially important for this work is the fact that an ACS does not create totally specialized rules at first and then tries to generalize them by e.g. genetic algorithms but specializes its classifiers for the present only in the *changing components*. Only when a classifier has got a mark, meaning that it was selected in an other state where the anticipated consequences were not correct, *unchanging components* are explicitly specialized, too, when entering again a successful state.

3 THE EXPERIMENT

As mentioned above the experiment about learning of the hand-eye co-ordination of a robot arm in conjunction with a camera was introduced by Birk (1995). A detailed description of this experiment is given now.

3.1 THE CODING

The camera (the eye) is vertically watching a black planar surface. The picture is diverted in a $n \times n$-matrix. Over this surface, the robot-arm with a red gripper is installed. It can move in the four directions North, South, East and West over the plain. Besides there is a blue block that can be gripped by the robot-arm and be moved around the planar surface. The robot-arm has got a feeling sensor that switches from 0 to 1 when it feels a block under it. That is why the possible environmental states can be coded in $S = \{black, red, blue\}^{n \cdot n} \times \{0,1\}$. The possible motor actions are R = {N = North, S = South, E = East, W = West, G = Grip, R = Release}. As mentioned Birk implemented this experiment also on a real robot. Therefore the coding was not changed. A schematic picture of an environment with n = 5 shows figure 3.

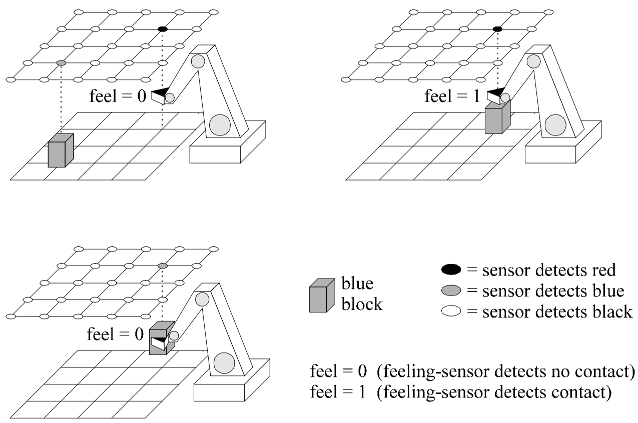

Figure 3: Task of learning to control a robot-arm via a camera-eye and one feeling sensor (cf. Birk, 1995, p. 35)

3.2 THE SIZE OF THE KNOWLEDGE

As in all Classifier Systems the knowledge is represented by its classifiers. The ACS has to generate in each position a classifier that moves in each direction with or without the block, onto the block or off the block. Besides in each position it needs a classifier that is able to grip and to release the block (in case the block is under it). Table 1 shows the necessary raw classifiers for a 2×2 matrix for each of this possibilities, differentiated only in the changing components.

Table 1: Sorted representation of the classifier list that consists of the knowledge generated without further specialization for *n*=2 (r: red; b: blue; s: black; N: north; S: south; W: west; E: east; G: grip; R: release)

move onto the block	**move off the block**	**move without contact to the block**
(b#r#0,N,r#s#1)	(s#r#1,N,r#b#0)	(s#r##,N,r#s##)
(#b#r0,N,#r#s1)	(#s#r1,N,#r#b0)	(#s#r#,N,#r#s#)
(r#b#0,S,s#r#1)	(r#s#1,S,b#r#0)	(r#s##,S,s#r##)
(#r#b0,S,#s#r1)	(#r#s1,S,#b#r0)	(#r#s#,S,#s#r#)
(br##0,W,rs##1)	(sr##1,W,rb##0)	(sr###,W,rs###)
(##br0,W,##rs1)	(##sr1,W,##rb0)	(##sr#,W,##rs#)
(rb##0,E,sr##1)	(rs##1,E,br##0)	(rs###,E,sr###)
(##rb0,E,##sr1)	(##rs1,E,##br0)	(##rs#,E,##sr#)
transport the block	**grip the block**	**release the block**
(s#b##,N,b#s##)	(r###1,G,b###0)	(b###0,R,r###1)
(#s#b#,N,#b#s#)	(#r##1,G,#b##0)	(#b##0,R,#r##1)
(b#s##,S,s#b##)	(##r#1,G,##b#0)	(##b#0,R,##r#1)
(#b#s#,S,#s#b#)	(###r1,G,###b0)	(###b0,R,###r1)
(sb###,W,bs###)		
(##sb#,W,##bs#)		
(bs###,E,sb###)		
(##bs#,E,##sb#)		

Therefore the size of the knowledge is n^2 each for gripping and releasing, $4(n-2)^2+3\cdot4(n-2)+2\cdot4=4n^2-4n$ for *transporting the block, moving without contact to the block, moving off the block* and *moving onto the block.* Because of the passthrough signs in ACS *moving without contact to the block* is independent from the actual position of the block and therefore has got the same size as *moving off the block*, *moving onto the block* or *transporting the block.* All in all that is $4(4n^2-4n)+2n^2=18n^2-16n$. Table 2 shows the size of the necessary knowledge for n=2 to 7.

Table 2: The number of different classifiers necessary for a complete knowledge

n=2	**n=3**	**n=4**	**n=5**	**n=6**	**n=7**
40	114	192	370	552	770

3.3 MORE SPECIFIC KNOWLEDGE

Considering the classifiers for transporting the block, one can realize that each of these classifiers can also match in a state where the arm and the block have no contact as each #-sign could either match with an 's' for black or with an 'r' for the robot arm. The same problem arises when looking at the classifiers for releasing the block. These classifiers have to be specialized in *unchanging components* (cf. section 2.4) to make sure that they are only applicable in the correct state. This is done by the marking mechanism.

Taking e.g. the first classifier for transportation in table 1: s#b## N b#s##, meaning that when transporting the block one step northwards the third component becomes black (s) and the first component becomes blue (b). It is applicable in the states ssbs0, srbs0 and ssbr0 but only successful in ssbs0, as in the other two states the robot is in a different position without the block. Once generated it will also be applied in the other states and will be marked there. Let us assume that the classifier has got a mark in state srbs0. Now it is applied again in the successful state ssbs0 and realizes that it has got the mark srbs0. That is why it will generate a new classifier that will explicitly carry an 's' (instead of the '#') in the second component to differentiate itself from the other state: ssb## N bss##. After setting another mark in ssbr0 and again being applied successfully it will be ssbs# N bsss# and will only match the correct state from now on.

Table 3: Sorted representation of the classifier list that consists of the complete and unique knowledge (after specialization) for *n*=2 (r: red; b: blue; s: black; N: north; S: south; W: west; E: east; G: grip; R: release)

move onto the block	**move off the block**	**move without contact to the block**
(b#r#0,N,r#s#1)	(s#r#1,N,r#b#0)	**(s#r#0,N,r#s#0)**
(#b#r0,N,#r#s1)	(#s#r1,N,#r#b0)	**(#s#r0,N,#r#s0)**
(r#b#0,S,s#r#1)	(r#s#1,S,b#r#0)	**(r#s#0,S,s#r#0)**
(#r#b0,S,#s#r1)	(#r#s1,S,#b#r0)	**(#r#s0,S,#s#r0)**
(br##0,W,rs##1)	(sr##1,W,rb##0)	**(sr##0,W,rs##0)**
(##br0,W,##rs1)	(##sr1,W,##rb0)	**(##sr0,W,##rs0)**
(rb##0,E,sr##1)	(rs##1,E,br##0)	**(rs##0,E,sr##0)**
(##rb0,E,##sr1)	(##rs1,E,##br0)	**(##rs0,E,##sr0)**
transport the block	**grip the block**	**release the block**
(ssbs#,N,bsss#)	(r###1,G,b###0)	**(bsss0,R,rsss1)**
(sssb#,N,sbss#)	(#r##1,G,#b##0)	**(sbss0,R,srss1)**
(bsss#,S,ssbs#)	(##r#1,G,##b#0)	**(ssbs0,R,ssrs1)**
(sbss#,S,sssb#)	(###r1,G,###b0)	**(sssb0,R,sssr1)**
(sbss#,W,bsss#)		
(sssb#,W,ssbs#)		
(bsss#,E,sbss#)		
(ssbs#,E,sssb#)		

Besides, the classifiers for moving without contact to the block have to be specialized, too, i.e. in the last (the feeler) component, to make sure not to be over the block.

Let's consider e.g. the first classifier for moving without contact to the block in table 1: s#r## N r#s##. It is applicable in the states sbrs0, ssrb0, ssrs1, but only correct in sbrs0 and ssrb0. So eventually it will be marked by ssrs1. Let's say it is successfully applied in sbrs0 again. Now it is possible to specialize the second or the last component. While the generated classifier that is differentiated in the second component matches from now on only in sbrs0, the generated classifier that is differentiated in the last component matches in both correct states and therefore is more useful than the other. Currently this is an unsolved problem in ACS. Despite this, eventually the classifier list will look like in table 3, with a complete and unique knowledge.

3.4 TESTING THE KNOWLEDGE

For monitoring purpose of the evolved knowledge a testing method of the currently achieved knowledge is necessary. The testing method is taking all the certain classifiers and checks if they are capable to solve all the tasks mentioned in 3.2. Although it should be irrelevant where the block is situated in the skill *moving without contact to the block* all block positions are tested for making sure, that all correct and unique classifiers are formed (see problem in 3.3). Meaning that once achieved a 100% test result all possible goals will be reached successfully and even in an optimal number of steps. Therefore the size of the tested knowledge is $2n^2$ for gripping and releasing, $n^2(4n^2-4n)$ for all possible movements without the block considering all possible block positions and $4n^2-4n$ for all possible movements with the block resulting in $2n^2+(4n^2-4n)\cdot(n^2+1)$.

Note the difference between the growth of the different action tests. While the complexity of the actions for the possible movements without the block is $\in O(n^4)$, the complexity of the other, actual more difficult actions is $\in O(n^2)$. Table 4 shows the growth of the sizes.

Table 4: The size of all tested actions and in comparison the size of the possible actions with the block

n	No. of all tested actions	No. of actions with the block
2	48	16 (32%)
3	258	42 (16%)
4	848	80 (9.4%)
5	2130	130 (6.1%)
6	4512	192 (4.3%)
7	8498	266 (3.1%)

4 ADDING ACTION-PLANNING TO ACS

In the introduced environment the probabilities of the occurrences of each state is not equal. Transporting the block is only possible if the system tried to grip the block while over the block. This will occur quite seldom as the latent learning process is not rewarded anywhere and gripping in any other position than the block position will result in weakening the strength of the grip-classifier. Transporting needs a successful grip before it can be executed so that it is still more unlikely.

As it will be shown action-planning is a possibility to decrease this effect.

4.1 THE ACTION-PLANNING PHASE

Prior to this ACS simply explore their environment in an latent learning phase. Now the new action-planning phase will be introduced.

When doing action-planning the C^{ant}-part of a classifier is used to calculate an anticipation of the next environmental state m^{ant}. It is assumed that m^{ant} is equal to the next m^{real} and so it is possible to calculate all new satisfied classifiers. For each new satisfied classifier this can be done again. The same process can take place backwards, too. That is how a bi-directional search is used to generate an optimal sequence of classifiers (at least the system "believes" it is optimal) with R-parts that lead from the actual state to a desired goal state. Action-planning is done without any interactions with the environment. This paper will not focus on the question which goal state is desirable but the goals will simply be generated externally by a so called goal-generator, that is part of the environment and not part of the ACS itself.

In the action-planning phase ACS requests a goal from the goal-generator. It tries to find the shortest possible path from the actual state to the goal by using the bi-directional search mechanism mentioned above. If it finds such a path, then it executes the classifiers of the path like in the exploration phase. If the execution was successful, then the ACS requests a new goal from the goal-generator. If the execution of any found sequence failed or if the search mechanism did not find a sequence at all, the ACS leaves the action-planning phase and reenters the normal exploration phase. Each 50 steps the action-planning phase is taking place.

4.2 THE APPLICATION OF ACTION-PLANNING IN THE EXPERIMENT

The goal-generator generates continuously the following goals:

1. Move to a different randomly selected place, where the block is not situated.
2. Move to the block and grip it.
3. Release the block at a different randomly selected position.

The goal-generator and the switching between the action-planning phase and the exploration phase is analogous to Birk (cf. 1995, p. 61ff).

Let's consider an example in which the ACS has already learned the complete world model (see table 3). Let's e.g. assume the following constellation of a 2x2 matrix: sbsr0, meaning the block is situated Northeast and the robot hand Southeast. Now the action-planning phase starts. The goal generator in phase one creates e.g. the goal sbrs0. The search mechanism will find a sequence that consists of only one classifier: ##sr0 W ##rs0. After the execution of this classifier the environmental state will be sbrs0 and the ACS requests a new goal. The goal generator, now in phase two, will generate the goal srss1 and after that sbss0. The search mechanism generates the sequence: ##sr0 E ##rs0, #b#r0 N #r#s1 or s#r#0 N r#s#0, rb##0 E sr##1 and after that #r##1 G #b##0. After the execution of both sequences the environmental state is sbss0 and the goal generator enters phase three and generates e.g. rsss1, that leads the ACS to the sequence sbss# W bsss#, bsss0 R rsss1. After the execution of these two classifiers the goal generator again enters phase one and the process starts from the beginning.

If the world model is not learned completely, then on the one hand a sequence might be longer than the optimal path and on the other hand the action-planning process might fail. In this case the ACS reenters the exploration phase.

4.3 ADVANTAGE OF ACTION-PLANNING

Due to the permanent request of the goal-generator during phase 2, to move to the block and to grip it, the system will be much more often in a position over the block and so experiences the results of gripping much more often. That is how the unequal distribution of probabilities between the different states shown in table 4 is partly neutralized. For there is a quadratic growth of the number of fields but the number of blocks stays the same (only one block) this advantage should increase with the size of the matrix.

5 RESULTS

This section on the one hand shows the positive effect of adding action-planning to ACS and on the other hand the overall performance of ACS in this simulation.

5.1 COMPARING ACS IN CONJUNCTION WITH AND WITHOUT ACTION-PLANNING

Figure 4 shows the mean results of 100 simulations of the $n \times n$ matrix for n=2 and n=3 without and with action-planning. The positive action-planning effect becomes obvious.

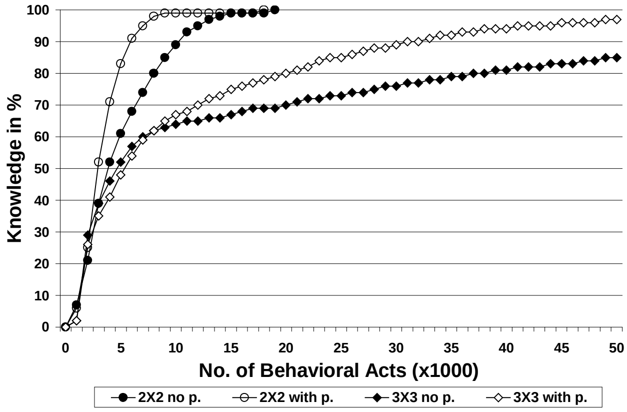

Figure 4: Results of the simulation with n=2, 3 without and with action-planning

Table 5 shows the same results in numeric form and the mean results of 40 simulations for n=4 without and with action-planning.

Table 5: Average results for n=2-4 with and without action-planning

	2x2 no plan	2x2 with plan	3x3 no plan	3x3 with plan	4x4 no plan	4x4 with plan
>75%	7200	4400	27200	15000	14500	16000
>80%	8000	4800	36400	19400	37500	24000
>85%	9000	5200	48200	24000	97500	49000
>90%	10400	6000	62400	30800	187500	75000
>95%	11800	6800	82000	40800	320500	89000
>99%	15000	8200	114800	61800	>600000	301000

Obviously the ACS with action-planning needs about half the behavioral steps of the one without action-planning. The effect is still increasing for increasing numbers of n. This confirms the reflections in section 4.3.

The result that 75% of the 4x4 matrix are learned faster than 75% of the 3x3 matrix is an effect of the measurement of the knowledge. As explained in 3.4 the complexity of the possible movements without the block is $O(n^3)$ while the one with the block is $O(n^4)$. During the exploration phase the movements without the block are taking place most of the time so that most of the corresponding classifiers are learned very quickly. For n=4 this is 90.6% of the world model and for n=3 only 84%. The same effect explains the slightly better performance of ACS without action-planning than the performance of ACS with action-planning for the 4x4-matrix in the 75% results because an ACS without action-planning is always in the exploration phase.

Figure 5 shows the results of 100 simulations each of a 3x3 matrix without and with action-planning in detail revealing the differences between the single experiments and again the positive action-planning effect.

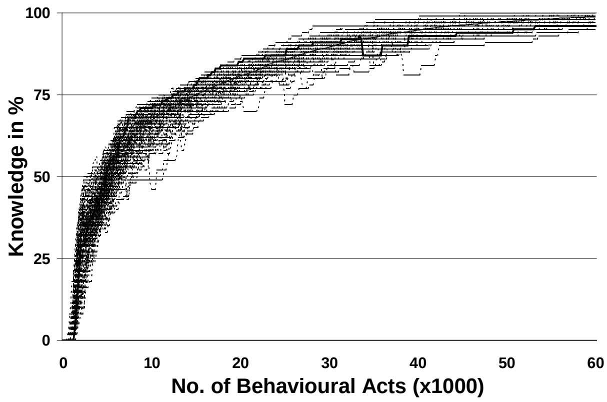

Figure 5a: 100 single experiments for action-planning in a 3x3 matrix with action-planning

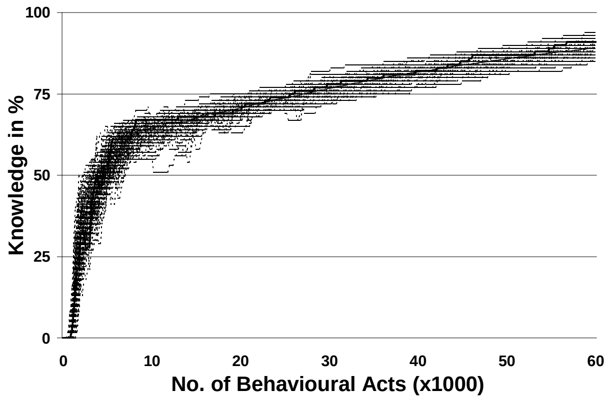

Figure 5b: 100 single experiments for action-planning in a 3x3 matrix without action-planning

5.2 MORE RESULTS WITH ACTION-PLANNING

As shown in 5.1 the performance of ACS in conjunction with action-planning is much better than without action-planning. Therefore some more results for $n \times n$ -matrixes with action-planning are calculated. Table 6 shows the average performance for n=4 (for reference purposes), 5-6 with 10 experiments each and for n=7 one example experiment is given.

Table 6: The results for n= 4 to 7 in the learning constellation with action-planning

	n=4	**n=5**	**n=6**	**n=7**
>75%	16000	16000	18000	18000
>80%	24000	23000	22000	24000
>85%	49000	42000	44000	30000
>90%	75000	99500	86000	54000
>92%	89000	141500	154000	80000
>94%	107500	194500	236000	158000
>96%	139000	276500	412000	428000
>98%	195500	479500	710000	1188000
>99%	261500	727500	1115000	1938000

5.3 COMPARISON WITH BIRK'S SRL

Birk (1995) developed a learning algorithm called "Stimulus Response Learning" (SRL) to solve the presented experiment. Analogous to Birk we were able to show the positive effect of action-planning, although his experimental results seem to prove an even greater effect.

But first of all Birk only refers to the time his Computer needed for the different tasks, so that a direct comparison is not possible. Besides his system is not creating simple rules but a graph, whose edges are not trying to generate common results but indeed monitoring the effects completely. Last but not least the knowledge about the environment is not complete as in ACS. The junctions in SRL are too common. That is why a gripping action is possible to take place where no block is under the robot-arm.

In the current system the possibilities of planning are restricted (...) the system is normally not able to generate a successful plan for gripping a block (Birk, 1995, p. 71, own translation).

These restrictions are not given in ACS. Once the system generated a complete world model it is not anymore limited and all possible tasks will be performed correctly.

6 CONCLUSIONS

Anticipatory Classifier Systems can generate plans. The examined experiment shows that ACS are able to take advantage out of generating plans to given goals even during the latent learning phase. It is able to learn much faster in environments where the probability of entering each possible state is not equal.

In this paper the goal generating process is part of the environment and not part of the ACS (cf. section 4.1). Future work may consist in integrating the goal generating process into ACS. A possibility would be to add a new memory to ACS which keeps a certain number of the last states in mind. ACS would be able to select goals out of these reminded states by e.g. selecting positions that were visited quite long ago. This approach can even be psychologically confirmed. In contrary to the known *lust of satisfaction* which could be compared to normal reward learning a so called *lust of function* is observable in humans, that describes the tendency to want to experience the correct outcome of anticipated behavioral consequences. This lust of function is especially observable in children when they want to do something over and over again (Hoffmann 1993). Obviously the effects of such self-goal-generating processes will have to be examined in detail.

References

Birk, Andreas (1995). *Stimulus Response Lernen*. Dissertation. University of Saarbruecken. Germany.

Forrest, Stephanie (1985). *A study of parallelism in the classifier system and its application to classification in KL-ONE semantic networks.* PhD Dissertation. University of Michigan.

Hoffmann, Joachim (1992). Probleme der Begriffsbildungsforschung: Von S-R Verbindungen zu S-R-K Einheiten. *Sprache undKognition*. 11(4) 223-238.

Hoffmann, Joachim (1993). *Vorhersage und Erkenntnis*. Göttingen: Hogrefe.

Riolo, Rick L. (1991). Lookahead Planning and latent learning in a classifier system. In Meyer, J.A., and Wilson, S.W. (editors). *From Animals to Animates: Proceedings of the First International conference on Simulation of Adaptive Behaviour*. Cambridge, MA: The MIT Press. 316-326.

Seward, John P. (1949). An Experimental Analysis of Latent Learning. *Journal of Experimental Psychology*. 39 177-186.

Stolzmann, Wolfgang (1997). *Antizipative Classifier Systeme*- PhD Dissertation. Fachbereich Mathematik /Informatik. University of Osnabrueck. Aachen: Shaker Verlag.

Stolzmann, Wolfgang (1998). Anticipatory Classifier Systems. In Koza; John R et al.(editors). Genetic Programming 1998. *Proceedings of the third Annual Conference, July 22-25, 1998, University of Wisconsin, Madison, Wisconsin*. San Francisco. CA: Morgan Kaufmann. 658-664.

Sutton, Richard S., and Barto, Andrew G. (1998). *Reinforcement Learning*. Cambridge: The MIT Press.

Tolman, Edward C. (1932). *Purposive behavior in animals and men*. New York: Appleton.