Adapted Pittsburgh Classifier System: Building accurate strategies in non markovian environments

Gilles Énée
Equipe systèmes distribués - GRIMAAG
Université des Antilles et de la Guyane
Campus de Fouillole B.P.592
97157 Pointe-à-pitre
genee@univ-ag.fr

Mathias Peroumalnaïk
Equipe systèmes distribués - GRIMAAG
Université des Antilles et de la Guyane
Campus de Fouillole B.P.592
97157 Pointe-à-pitre
mperouma@univ-ag.fr

ABSTRACT

This paper focuses on the study of the behavior of a genetic algorithm based classifier system, the Adapted Pittsburgh Classifier System (A.P.C.S), on maze type environments containing aliasing squares. This type of environment is often used in reinforcement learning litterature to assess the performances of learning methods when facing problems containing non markov situations.

Through this study, we discuss on the performances of the APCS on maze type environments and also of the efficiency of an improvement of the APCS learning method inspired from the XCS : the covering mechanism. We manage to show that, without any memory mechanism, the APCS is able to build and to keep accurate strategies to produce regular suboptimal solutions to these maze problem. This statement is shown through a comparison of the results obtained by the XCS, XCSM and XCSMH on distinct maze problems with these obtained by the APCS.

Categories and Subject Descriptors

I.2.6 [**Learning**]: [Concept learning, knowledge acquisition]

General Terms

Algorithm, Performances, Verification

Keywords

APCS, XCS, classifier systems, non-markovian multi-step environments, strategy

1. INTRODUCTION

Classifier systems based on genetic algorithm are rule based systems whose diagnose ability is known to be used on parameters optimisation problems [7].Nevertheless, before being used in a production and diagnostic context, this type of classifiers needs to perform a learning step. Most often, this learning stage is performed on a sample of data representing the available and validated / expertised data of the considered environment.

Tendencies contained by this data set are assimilated by the classifier system using reinforcement learning : the system is continuously exposed to signals created using the learning sample. At this point, the action performed by the classifier, in reaction to the incomming signal, is used to improve the accuracy of its answer thanks to the fitness function. This function is defined depending on the learning problem considered: its aim is to maintain strong classifiers within the population by preventing them to be deleted or lost by genetic pressure.

In the literature, we encounter various methods used to successfully perform this reinforcement learning but, concerning learning classifier system using genetic algorithms, Q-Learning reinforcement methods and anticipation based methods are the most widely used [11, 3, 2, 12].

In order to characterize the efficiency of a learning method performed by a given learning classifier system in front of a given reinforcement learning problem, we need to find a test environment that offers an appropriate set of measure tools.

To fulfill this requirement, we introduce in Section 2 the type of environment chosen to perform these measures with a precise description of these used in this study. Then, in Section 3, we describe the main algorithm of Adapted Pittsburgh Classifier System, including its latest improvement. After this point, we will describe the experiments we conducted and discuss the obtained results in Section 4. This discussion will be extended to Section 5 with a comparison of the results previously obtained with results obtained with the eXtented Classifier System (XCS [3]) on the same reinforcement learning problems and with classifier systems which are known to solve POMPD problems in order to discuss of the pertinency of the obtained results. We will then conclude on the measured improvements led by the covering mechanism and on ongoing experiments.

2. CONTEXT AND RELATED WORK

2.1 The animat problem

Maze problems, as simplified reinforcement learning problems, are often used in classifier systems literature to assess the efficiency of a learning method (XCS, ZCS), an improvement of an existing classifier system (XCSM, ZCSM), or to validate a new algorithm (ACS, AgentP, ATNoSFERES)

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

[1]. Moreover, some mazes also offer perceptually similar situations that requires different actions to reach the goal: these situations are known as "aliasing squares". The abilities needed by a learning classifier system to solve a maze problem can also be easily related to the abilities needed to solve a given optimization problem with a learning sample containing missing or aliased data.

Most often, a maze is defined as a given number of neighboring cells. A cell is a bounded space formally defined: it is the elementary unit of a maze. When it is not empty, a given cell of a maze can contain an obstacle, food, an animat and eventually a predator of the animat.

The maze problem is defined as following: an animat is randomly placed in the maze and its aim is to set its position to a cell containing food. To perform this task, it possess a limited perception of it environment: it can only see the eight cells surrounding its position. The animat can also only move to an empty cell between these neighboring cells, moving step by step through the maze in order to fulfill its goal.

The problem given to the cognitive system which behavior is studied on these environment is to attend to adopt a policy inside this environment. This strategy must allow the animat to complete its goal using a minimal number of steps.

Maze environments also offer plenty of parameters that allows to evaluate the complexity of a given maze and also to evaluate the efficiency of given a learning method. The full description of these parameters is available in [1].

The following chart[1] (fig. 1) presents most of mazes that are available in the literature. It was built considering the type of the aliasing squares contained by each maze and by realizing the mean of the average number of steps ϕ_m^Q done by a Q-Learning algorithm to reach the food over the average distance to the food $\phi_m$measured for this maze.

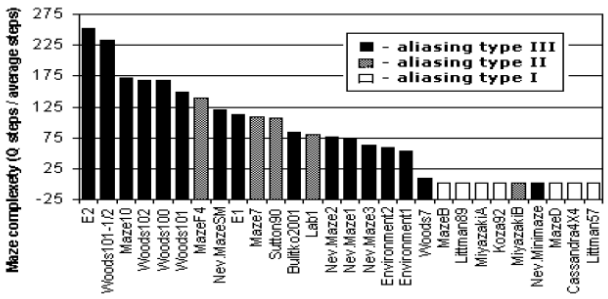

Figure 1: Complexity chart of maze-type environments

The Woods101 maze which is easy to represent (fig. 2) possess a high $\frac{\phi_m^Q}{\phi_m}$ rate ($\phi_m^Q = 402.3$ and $\phi_m = 2.7$, $\frac{\phi_m^Q}{\phi_m} = 149$ [16]). For the same reasons, two other mazes also raise our interest: the E2 ($\phi_m^Q = 710.23$ and $\phi_m = 2.33$, $\frac{\phi_m^Q}{\phi_m} = 304.81$ [16]) and the Maze10 ($\phi_m^Q = 890.83$ and $\phi_m = 4.56$, $\frac{\phi_m^Q}{\phi_m} = 195,35$ [16]).

As the E2 environment appears to be the most difficult maze, we will study the effects of the parameters proper to the APCS only on this environment. Woods101 and Maze10 will be used on the two last sections when comparing the results of APCS with results obtained with XCS, XCSM and XCSMH.

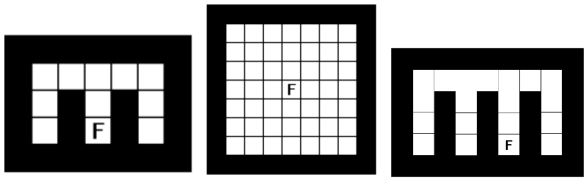

Figure 2: Woods101, E2 and Maze10 mazes

2.2 Random walk and optimal number of steps

In order to determine the quality of the obtained results on a given maze, we need to establish clearly two datas : the measure done using a random method and the results that would be obtained using optimal choices.

We have chosen to collect the same data on the random walk experiments, on the optimal choice approximation and on the experiments conducted with the classifier systems. Concerning the random walk, the animat is randomly placed in the maze and at each time step it chooses a random direction between the eight directions available. We measure the number of steps done by the animat and the final distance of the animat to the food (this choices are clearly established in Section 4.2, please refer to it for more details).

When considering the optimal number of steps that should be done by the animat to fulfill its goal, we had to consider the fact that the mazes we have chosen contain aliasing squares. As a consequence, we have chosen to refer to previous work of Zatuchna[16] on MASS and AgentP when dealing with measures of optimal performances for the considered mazes.

3. PRINCIPLES OF STUDIED CS

3.1 The Adapted Pittsburgh Classifier System

The Adapted Pittsburgh Classifier System (APCS) differs from other kind of classifier systems on many points. We will review in the next sub-sections how does structure, evaluation and evolution mechanism differ from other classifier systems and from Smith' original work[13].

3.1.1 Structure

APCS is built with production rules also called classifiers. These classifiers are split into a condition part (also called sensor) that reads the environment signal and an action part (also called effectors) that acts upon the environment. Usually, the condition part is defined upon a ternary alphabet $\{0, 1, \#\}$, where # replaces 0 and 1, it is also called *wildcard*. The action part contains only bits. Other representations are possible but as we will not use them, we will not describe them further. Classifiers are melt together to form an individual i.e. an individual is a set of classifiers also called *knowledge structure* or *knowledge base*. Finally, a population of an APCS is filled with several individuals.

The internal language proposed by [14] to instantiate variables within condition part is not implemented in APCS. So a population is initially created using four parameters:

- A fixed number N_i of individuals in population.
- A fixed number N_c of classifiers per individual.
- A fixed size L_c for all classifiers.
- An allelic probability $P_\#$ of having a wildcard in the condition part.

A population is first filled with random classifiers. It is also possible to fill initial population with chosen classifiers. We will focus now on how APCS' population is evaluated.

3.1.2 Evaluation mechanism

Individuals interact with the environment through sensors and effectors. They are rewarded thanks to a multi-objective fitness function. Thus, individuals have a strength that globally reflects the mean strength of the classifiers filling it. The specific structure of APCS implies a specific evaluation mechanism. Individuals are evaluated at once i.e. with parallel firing classifiers. This is due to the fact that individuals are potential solutions to the problem so, they are not linked to each others as in a Michigan CS (see [5]).

Individuals are composed of classifiers: in order to ensure that all classifiers had a chance to be tested before evolution occurs, individuals must have been tested a number K of *trials*. A trial consists in three steps (see fig. 3). First, the environment sends a signal to the individual currently evaluated. Then, thanks to that signal, the individual fire a classifier that matches the incoming signal. The triggered classifier produces an action upon the environment. Last, the fitness function gives a reward to the whole individual for the effector fired by one classifier. As a consequence, we need to set K over $\frac{N_c}{2}$[5] to ensure that every single classifier had at least one chance to be evaluated.

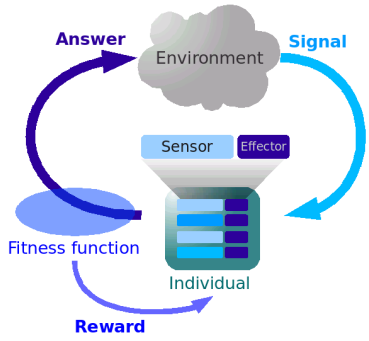

Figure 3: Evaluation mechanism

The evolution algorithm (here genetic algorithm) is applied when all individuals had been evaluated K times. In this case, the reward can be continuous, using a Q-Learning method for fitness / strength update, or it can be reset at each generation and based only upon the last generation trials.

When an individual is evaluated, many classifiers can match the environment signal. Smith proposed to give a relative weight to each action of matching classifiers. These weights would be used as a probability of choosing a particular effector. This mechanism can be found in XCS with a more sophisticated evaluation of effector weights. However, action or classifier weight is not implemented in APCS.

As shown in this description, individuals are evaluated independently. Thus, each individual possesses its own environment that allows it to be evaluated independently from other individuals. This aspect of the APCS also permits to evaluate individuals in parallel. The major interest for separated environment is that the genetic algorithm should mix together positive experiences of each individual.

We will now describe the evolution mechanism.

3.1.3 Evolution mechanism: The Genetic Algorithm

Genetic Algorithm (GA) is essential to APCS: it allows learning and evolution within the classifier system. The GA applies its four main operators among individuals of the population using their fitness. It first **selects** parents that will eventually reproduce using **crossover** operator and uses **mutation** operator to create new offspring that can also be **inverted**.

The selection mechanism uses mainly the roulette wheel or the tournament methods to select parents that will be kept to generate next generation.

The crossover operator chooses randomly a $\{n, i\}$ pair where $i\epsilon\{2, L_c - 1\}$ is the position where crossover will occur within classifier n ($n\epsilon\{1, N_c\}$) in each individual selected for reproduction. The crossover is thus one point and manipulates individuals of the same size. The reason why Énée has chosen fixed length individuals [5] comes from Smith' work[13]. Smith has proved that the crossover operator between individuals of different lengths will mainly accentuate the difference of length between the two offspring's. The selection mechanism would then progressively erase short individuals that would not be able to answer problems because they do not have enough classifiers. Thus, bigger individuals would be selected and the individual size would tend to grow through generations. Smith also observed that while the individuals tend to grow, they first reach an optimal size to well answer to the problem. Then additional classifiers appear and produce noise in answer. Finally, without individuals and classifiers of fixed length, the Pitt-CS becomes unable to reach an optimal solution. Thus, having an infinite cognitive capacity[1] to answer a question is useless assuming that the considered problem can be solved with a finite cognitive capacity[**?**].

Mutation operator is allelic. Crossover and mutation and inversion are expressed as probabilities.

The inversion operator can be used with APCS because each classifier is member of an unordered knowledge structure: an individual. So inversion of classifiers within an individual allows classifiers to be better exchanged when single point crossover occurs.

Best genetic algorithm parameters usually taken by Smith are (except for the selection mechanism):

- Selection mechanism: roulette wheel.
- Crossover probability: 50-100%.
- Allelic mutation probability: 0,01-0,5%.
- Inversion probability: 50%.

The inversion operator is not used by the APCS in order to allow the system to make a supra-individual emerge within its population[6]. This "unique" individual appears thanks

[1] N_i or N_c infinite

to the genetic algorithm. As the inversion operator changes the order of classifiers within individual, it would prevent this "supra" individual to emerge.

3.1.4 Further discussion upon APCS and covering mechanism

When a signal comes from environment, several classifiers can match this signal. This classifier selection problem can be solved for example :

- by taking randomly one classifier among the matching ones,
- by taking the most specific classifier i.e. with the lowest number of #,
- or by taking the most generic classifier i.e. with the greatest number of #.

For the last two methods, if several classifiers still match, the random selection mechanism is used among the selected matching classifiers. This permits to have different actions for one signal. Thus, APCS should be able to find non markovian processes to solution.

The simulator of the environment that is used to make the population learn can be used in two ways. It can be reset at each generation or set to its previous status using a learning algorithm. The last method is not within the original framework of Smith where system is GA dependent only for learning. Parallelism can be easily simulated through a simulator that keeps track of its status for each individual at the end of previous evaluation. If the simulator is reset, parallelism can be easily translated in sequential evaluation of individuals also.

To have a clear and complete view of the evaluation mechanism, see algorithm 1.

The ending criteria can be a choosen number of generation or when a choosen number of individual receive maximal rewards for each one of the K trials. So to evaluate an APCS, you need these parameters:

- Simulator / Environment reset mode: To zero / To previous status.
- A number of trials.
- A selection rule mechanism: random / specific / generic.
- A number of generations.

Covering mechanism is newly implemented in APCS. Directly inspired from Wilson work with XCS[15], it consists here in replacing the sensor part of a classifier when no classifier matches the signal from environment. To enhance this mechanism, we add a parameter to each classifier, called covering time C_t, in order to measure the number of generations a classifier has not been activated before it should be covered. As a consequence, each classifier has a chance to bring strong genomic precursors to collectivity, i.e. individuals, before being covered. The covering mechanism replaces the condition part of a classifier with the message from environment adding wildcards thanks to the wildcard probability $P_{\#}$ found in genetic algorithm parameters. Thus the action part is kept as it was before covering.

We have made a quick technical tour of APCS, let's have a quick tour of XCS.

Algorithm 1 of Adapted Pittsburgh style classifier system

```
Begin
  Fill(P); { Random initialization of P or using
a file initialization.  }
  Generation = 1;
  Repeat
    For (All I_j of P) Do
      Reset_Environment(j); { Reset environment to
individual I_j last status or reset it to zero.  }
      Reward_{I_j} = 0; { If reward is continuous,
then you should remove this line.  }
      For (A number of Trials k) Do
        Fill(Message); { with message from envi-
ronment.  }
        Fill(Match-List); { Create Match-Set from
I_k classifiers that match signal.  }
        If (IsEmpty(Match-List)) Then
          ReplacePosition = ChooseCoveredClassi-
fier(); { Select a classifier not used since C_t
last generations.  }
          CoverClassifier(ReplacePosition, Mes-
sage); { Replace condition part of classifier num-
ber ReplacePosition with Message and wildcards.
}
        Fill(Match-List);
        Endif
        If (Size(Match-List)> 1) Then
          C = BestChoice(Match-List); { Choose
classifier in Match-List using strategy described
in this section.  }
        Else
          C = First(Match-List);
        Endif
      A = Action(C);
        DoAction(A); { Acts upon environment using
action A.  }
        Reward_{I_j} = Reward_{I_j}+ ActionReward(A)/k; {
ActionReward is the fitness function.  }
      EndFor
      ChangeStrength(I_j,Reward_{I_j}); { Change indi-
vidual I_j strength using Reward_{I_j}.  }
    EndFor
    ApplyGA(P); { Genetic Algorithm is applied
after every individuals had been evaluated.  }
    Generation = Generation + 1;
  Until (Ending Criteria encountered);
End
```

3.2 The eXtended Classifier System

For this experiment, we compare the performances of the APCS with these obtained with an eXtended Classifier System on the same maze.

The XCS, which was first built by Wilson in 1995 [15] started to become mature near 2000 thanks to Butz, Lanzi and Kovacs who realized rigorous performance and accuracy studies over this system, allowing it to evolve through new mechanisms: improvements on the main algorithm, adding memory (XCSM1, XCSM2 [9]). As a consequence, this system show pretty good results, even on maze-type environments with aliasing squares (see [11]).

The XCS principle is quite easy to understand: it consists in a population of classifiers (sensor plus effector) called individuals, which evolution relies on the ability of each classifier to predict, thanks to a Q-Learning algorithm, the obtained reward for a given action. These classifiers are mixed with a classical GA and converge easily towards quite generalist classifiers that answer globally to the problem.

To realize our measures, we have used the version 1.2 of the XCS [4], which is related to the algorithmic version of XCS published in the article of Butz and Wilson [3]. This version includes most of the mechanisms developed until here for the XCS, without including register memory mechanism.

4. EXPERIMENTS - RESULTS

4.1 Experimental settings

For each experiment, we submitted 20000 consecutive problems to the system: for each problem, the animat is randomly put on a free square of the maze and the trial stops when one of these two conditions is fulfilled:

1. the position of the animat in the maze is equal to the position of the food
2. the number of steps done by the animat surpass a certain threshold (MaxSteps, equal to 50 steps for every presented results)

When the problem is solved, we record the starting distance to the food of the animat, its final distance to the food[2] and its total number of steps. Both APCS and XCS results shown in this paper are averaged over 10 experiments.

As done in [3] and in [11], the signal received by the system consists in a 16 bits string that represents each of the 8 squares surrounding the animat. These squares are encoded clockwise, starting by North: (00) stands for an empty cell, (11) for food and (10) for an obstacle. As a consequence, the sensor part of the classifier also contain 16 positions. Each position in the sensor can be randomly occupied by 0, 1 or by a wildcard (#). The effector part, coded by a string of 3 bits, stands for one of the eight directions available for the animat, coded clockwise, as the sensor part.

Specific settings used for XCS are the same as these used by Lanzi in 1999 [11], please refer to this experiment for more details.

Concerning the APCS, each evaluation group controls an animat. As a consequence, during the experiment, each group is submitted to the 20000 problems and solve them asynchronously. The experiment stops when all evaluation groups have solved at least 20000 problems. The number of moves is measured during each of the K trials (see section 3.1): if it is correct, (i-e if the animat moves toward an empty cell), the evaluated individual receives a reward of $\frac{0.2}{K}$ and the movement is performed by the animat of the group; if it reaches the food, it receives a reward of $\frac{1.0}{K}$ and the animat of this group is randomly replaced in the maze. Else, the individual receives a negative reward of $\frac{-0.5}{K}$ and the animat of the group is not moved.

[2]due to MaxSteps, it may be greater than 0

Concerning the GA step, the mutation mechanism allows to reinforce the exploratory ability of the system by creating new classifiers. These classifiers are created when modifying locally and randomly existing classifiers according to a certain rate, which is expressed by the chosen mutation probability P_{Mut}. However, two consecutive positions of the considered mazes rarely differs one from another than more than 4 bits, and the non-sense sequence (01) can, in the present case, invalidate the activation of a classifier. As a consequence, the more the number of mutated bits is high, the more the system may lose classifiers that could have allowed it to manage to find the food. Experiments performed in [5] have also validated that a high mutation rate ($P_{Mut} > 0.2$) prevents the system from keeping optimal classifiers.

The other important parameter of the GA step, the cross-over rate, has a great influence on the homogeneity and on the stabilization of the system: combined to elitism, it allows the system to preserve the best behaviors expressed inside the population. Énée had measured in [5]that a low cross-over rate ($P_{Cross} < 0.6$) slows the convergence of the system by preventing good genetic precursors to be replicated inside the population.

As shown in [6, 5], stable results are obtained using a mutation probability P_{Mut} set to 0.005 and a cross-over probability P_{Cross} set to 0.75. As a consequence, we have chosen to use these values of parameters to perform our experiments.

4.2 Results without covering

For the presented experiments, we made two significant measures: the first measure, the average number of steps done by the animats, allows us to tackle the ability of the system to conform to a moving policy inside of the maze. Due to that fact, the evolution of this measure reflects and characterize the evolution of this ability.

As a second measure, the average final distance of the evaluation groups at the end of a trial allows us to validate the results outlined by the first measure. As the animat is randomly replaced when the number of steps done crosses a certain threshold, this second measure shows the efficiency of the policy built by the system to allow its animats to achieve their goal.

	Maze E2	
	Avg. nb of steps	Avg. final dist.
Random walk	32.49	0.78
$N_I = 20$	24.84	0.58
$N_I = 30$	15.03	0.11
$N_I = 40$	13.43	0.08
$N_I = 50$	12.62	0.06

Table 1: Measure of the influence of the variation of the number of individuals on the average number of steps ($N_c = 30$, $N_I = 20, .., 50$)

Now, we can study the influence of parameters proper to the classifier system. The number of individuals and the number of classifiers are the only parameters that really have an influence on the cognitive properties of the CS ([5, 13]).

First, for a fixed number of classifiers ($N_c = 30$) , lets compare obtained results for N_I between 20 and 50 (Table 1).

When the number of individuals changes, it also modify the number of evaluation groups. As in this experiment, each group controls the moves of an animat, the classifier system is able to learn on N_I different situations for each trials. As a consequence, the raise of the number of individuals also raise the exploratory ability of the system which accelerates and improves the convergence of the system.

When considering the evolution of the average final distance to the food (Table 1, column "avg. final dist")., we can conclude that the raise of the number of individuals contribute to the system's stabilization and to raise it uniformity.

Lets now consider the influence of the variation of the number of classifiers contained by an individual on the performances of the system on this problem. The following experiments (Table. 2) have been conducted with a fixed number of individuals ($N_I = 30$) and various number of classifiers ($N_c$between 20 and 50).

Each classifier determine the answer of the individual to one or several given signals coming from the environment. As a consequence, the number of classifiers contained by an individual can possibly have an influence on the number of signals which may trigger an answer from a given individual.

	Maze E2	
	Avg. nb of steps	Avg. final dist.
Random walk	32.49	0.78
$N_c = 20$	33.35	1.25
$N_c = 30$	15.03	0.11
$N_c = 40$	12.53	0.06
$N_c = 50$	11.88	0.07

Table 2: Measure of the influence of the variation of the number of classifiers on the average number of steps ($N_I = 30$, $N_c = 20, .., 50$)

As shown by Smith [13], if the potential information contained by an individual is too high, the exceeding information generate noise that disturbs the answer of the system and the evolution of more fitted classifiers. In addition to this phenomenon, as non markovian situations trigger a higher wildcard rate in the classifiers [1], each additional classifier may bring more unnecessary information. This side effect emphasize the fact that it exists a potential information threshold on information contained by an individual. When considering the obtained measures, we can conclude that this threshold depend on the considered problem.

As a conclusion, if in the beginning, providing to the individuals of the APCS additional cognitive capacity can improve the quality of the answer of the system, additional classifiers may contain useless precursors that disturb the convergence of the system.

4.3 Results obtained using covering

In order to improve our results, we have adapted the covering mechanism used in the XCS to allow APCS to generate well fitted classifiers when encountering an unknown signal (Section 3.1.4). To measure the impact of the activation on the evolution of the classifiers contained by the individuals of the APCS, we have chosen to measure the changes registered when choosing different values for the C_t parameter. To prevent any additional perturbation, these tests have been performed on a fixed number of classifiers and a fixed number of individuals. In this paper, we have chosen to present these results for $N_I = 40$, $N_c = 30$ for the maze E2 with a C_t between 0 (no covering) and 30 (uses classifiers not triggered during 30 generations).

	Maze E2 ($N_I = 40$, $N_c = 30$)	
	Avg. nb of steps	Avg. final dist.
$C_t = 0$	13.43	0.08
$C_t = 3$	8.65	0.01
$C_t = 7$	8.47	0.01
$C_t = 11$	8.60	0.01
$C_t = 15$	8.64	0.01
$C_t = 20$	8.48	0.01
$C_t = 30$	8.83	0.01

Table 3: Measure of the influence of the variation of the parameter C_t on the obtained results

We can notice on Table. 3 that the number of steps done by the system to reach the food decreases when increasing the C_t parameter. This improvement occurs due to the artificial modification of the available pool of classifiers by the covering mechanism. As we have measured, a classifier that have not been activated on C_t generations have strong probability to contain one or many defective precursors. As this mechanisms allows to remove these precursors from the population, it also increases the accuracy of the strategy built by the system. This phenomenon is suggested by the evolution of the average final distance to the food of the animats : the tendencies observed on the first measure are assessed by the second one.

As a summary, through the results presented on tab 3, we deduce that the classifiers with the condition part replaced by the covering mechanism carry an amount of information whose usability decrease depending on the raise of the number of evaluation steps passed without being activated. As the animat problem is studied on a multi step environment, it may occur that a classifier that have not been triggered at a given evaluation step t_i can be triggered at the step t_{i+n} depending on the moves of the animats through the environment. However, the number of squares which can be occupied by the animat are finite so it exist an upper boundary to the number of available signals generated by this environment.

Figure 4 represents the average over all the conducted experiments of the gain we register when we increase the C_t parameter. Due to the fact that this gain becomes equal to 0 when a certain value of C_t is crossed, we can suppose that it exists a finite number of generations N_{CT}which may allow us to diagnose that if a classifier has not matched any signal emitted by the environment during the last N_{CT} generations, the signals corresponding to this classifier are not available in the current environment. Moreover, according to the structure of the APCS (see Section 3.1), and if we

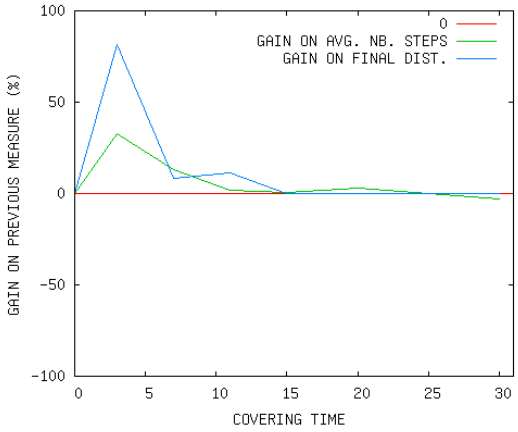

Figure 4: Evolution of the gain related to the covering mechanism on Maze E2

extend the range of this conclusion, we can also suppose that for each environment of this type, it exists a threshold value of evaluation steps $K * N_{CT}$ over which we can decide that an upper value of the C_t parameter does not carry any additional knowledge on the potentially useless information carried by a classifier.

5. DISCUSSION ON RESULTS

As shown by the experiments presented in this paper, APCS manage to evolve classifiers allowing it to adopt a stable policy whose quality is greatly improved by the used covering mechanism.

We will now focus on the comparative study of the best results we obtained with the XCS with the best results obtained with the APCS on the environments Woods101 ($N_I = 30$, $N_c = 20$ and $C_t = 7$), E2 ($N_I = 40$, $N_c = 50$ and $C_t = 11$) and Maze10($N_I = 40$, $N_c = 50$ and $C_t = 7$) (see tab. 4).

			XCS		APCS	
	Rand	Opt	steps	fdist.	steps	fdist.
Woods101	29.19	2.90	7.89	0.02	5.63	0.01
E2	32.49	28.20	19.74	0.18	7.86	0.01
Maze10	42.27	5.11	40.69	3.8	23.81	1.08

Table 4: Best average results (number of steps and final distance to the food) obtained by XCS and APCS on the considered environments, average number of steps obtained using random walk and optimal number of steps given by MASS

Parameters used for the XCS during these experiments are these used in the experiment conducted by Lanzi in 1999 on this type of environment[11], exception made for the number of individuals which is 2000 for the Woods101, 8000 for E2 environment and 10000 for Maze10 environment.

When we consider the differences between the best results obtained by the XCS and the best results obtained by the APCS, we notice two differences. The first one, significant on E2 and on Maze10 but not on Woods 101 is that the average number of steps done by the APCS is closer to the optimal than the average number of steps done by the XCS. This statement can find its foundation in the second difference noticed: the average final distance to the food for the APCS on the three mazes is lower than the one measured for the XCS. This difference means that the APCS accurately find the food more often than the XCS which implies that the policy built by the APCS is more stable and accurate.

However, the strategies employed by the two systems are quite different one from another: the XCS learning stage focuses on the value function and the policy deployed by this system is strongly dependent of this value function. In addition to that fact, in order to keep the accuracy of its prediction, this mechanism requires to maintain all the actions available for a given signal. Due to these facts, even if the XCS succeed in keeping a stable policy for the Woods 101 environment, it fails when facing an environment with numerous aliasing situations.

,

	XCS	XCSM	XCSMH	APCS
Woods101	7.89	3	2.9	5.63
E2	19.74	—	—	7.86
Maze10	40.69	15.1	6.1	23.81

Table 5: Comparison of the average number of steps done by the APCS on the considered mazes with performances obtained using LCS able to solve POMDP

As a consequence, Lanzi developped XCSM and XCSMH allowing XCS to use an inner register to keep a trace of its previous actions within its population. This internal register allows XCSM and XCSMH to deal successfully with maze environments containing aliasing squares by adding information to solve the aliased situations depending on the values contained by its internal register (please refer to [9, 10] for more details).

On the opposite, the learning mechanism used by the APCS relies on its cognitive capacity (N_c) which allows it to keep different efficient actions for each aliasing situation with the same probability of being chosen thanks to the random selection of matching classifiers.

The strongest classifiers will tend to stay in the population because they will allow their owner to reach the food more accurately and more often than the other individuals of the APCS. As a consequence, in a multistep environment, the classifiers contained by these strong individuals allow them to build action chains which are reward dependent. Moreover, instead of solving one problem at once, the APCS tries to solve N_I problems at the same time (see Section 3.1.2) which enables it to explore a higher number of situations in a short time of simulation.

Due to all these facts, the system tend to converge to a "sub-optimal" (see tab. 5 [8] [16]) solution that allows it to be rewarded accurately and more often without using anticipation or memory mechanisms.

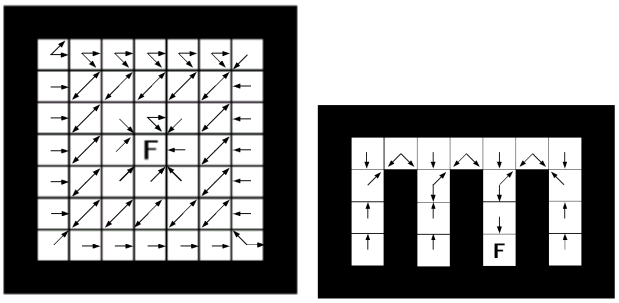

Figure 5: Example of policies deployed by the APCS to solve E2 and Maze10

We assist to the formation of policies (see fig. 5) that allows it to reach the food from every position, even when the environment contain aliasing squares. This policy evolves with the frequency of the reward encountered by the individuals which allow them to adapt and modify (via the genetic algorithm) their classifiers.

6. CONCLUSIONS

Through this paper, we have shown and studied results which indicate that, without any knowledge of its environment, even when facing non-markovian positions, the Adapted Pittsburgh Classifier System improved with the covering mechanism is able to adopt a quasi-stable policy in maze environments containing aliasing squares. This policy allows it to reach accurately the food with a low but not optimal number of steps as we have seen through a comparison with classifier systems using memory mechanisms.

When studying the number of classifiers contained by an individual, we have shown that the raise of this local cognitive capacity can benefit to the system if it remains under a problem dependent threshold (see also [6]). Cognitive capacity provided over this threshold shown the conservation of defective precursors and a strong disturbance of the system answer due to them. Fortunately, as shown in the experiments, these precursors are assimilated/eliminated by the system during a period depending on the useless amount of information they carry.

We have also shown that the covering mechanism we propose had a noticeable influence on the performances of the system: classifiers eliminated by this mechanism would carry an information that have a decreasing usability depending on the number of evaluation steps during which these classifiers are not triggered by a signal from the environment.

Some interesting further work remains to be finalized, especially concerning the precise built of the policy evolved by the APCS along the experiment and the role occupied by the number of evaluation trialsin the constitution of this policy. Further interesting work should also been conducted on the possible use of these properties on aliased data samples as available in economy or physics.

7. ACKNOWLEDGEMENTS

The authors would like to thank the Interreg IIIB project (section Prévison des Risques Majeurs) for funding and support. Thank also to both of the reviewers for their precious advices and corrections.

8. REFERENCES

[1] A. J. Bagnall and Z. Zatuchna. On the classification of maze problems. In L. Bull and T. Kovacs, editors, *Applications of Learning Classifier Systems, Studies in Fuzziness and Soft Computing, Vol. 183*, pages 307–316. Springer, 2005.

[2] L. Bull. Lookahead and latent learning in ZCS. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 897–904, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[3] M. Butz and S. W. Wilson. An algorithmic description of XCS. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *IWLCS*, volume 1996 of *Lecture Notes in Computer Science*, pages 253–272. Springer, 2000.

[4] M. V. Butz. Documentation of XCS+TS c-code 1.2. IlliGAL Report 2003023, Illinois Genetic Algorithms Laboratory, October 2003.

[5] G. Énée. *Systèmes de Classeurs et Communication dans les Systèmes Multi-Agents*. PhD thesis, Université de Nice, janvier 2003.

[6] G. Énée and P. Barbaroux. Adapted pittsburgh-style classifier-system: Case-study. In *IWLCS*, pages 30–45, 2002.

[7] G. Énée and C. Escazut. Evolution of communication in a genetic based multi-agent system: Use wise resources. In *Congress on Evolutionary Computation (CEC)*, pages 2038–2044, Portland, Oregon, June 20-23 2004. IEEE Press.

[8] J. H. Holmes, P. L. Lanzi, W. Stolzmann, and S. W. Wilson. Learning classifier systems: New models, successful applications. *Inf. Process. Lett.*, 82(1):23–30, 2002.

[9] S. Landau, O. Sigaud, and M. Schoenauer. Atnosferes revisited. In *GECCO*, pages 1867–1874, 2005.

[10] P. L. Lanzi. Adding Memory to XCS. In *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC98)*. IEEE Press, 1998.

[11] P. L. Lanzi. An analysis of the memory mechanism of XCSM. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 643–651, San Francisco, CA, USA, 22-25 July 1998. Morgan Kaufmann.

[12] P. L. Lanzi and S. W. Wilson. Optimal classifier system performance in non-markovian environments. Technical Report 99.36, Illinois Genetic Algorithms Laboratory, Milan, Italy, 1999.

[13] O. Sigaud. *Les systèmes de classeurs: un état de lárt*, volume 21 of *Revue d'intelligence Artificielle RSTI série RIA*. Lavoisier, 02 2007.

[14] S. F. Smith. *A Learning System based on Genetic Adaptive Algorithms*. PhD thesis, University of Pittsburgh, 1980.

[15] S. F. Smith. Adaptive learning systems. *Expert Systems: Principles and Case Studies*, 1984.

[16] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):148–175, 1995.

[17] Z. V. Zatuchna. *AgentP: A Learning Classifier System with Associative Perception in Maze Environments*. PhD thesis, School of Computing Sciences, UEA, 2005.