Imitation guided Learning in Learning Classifier Systems

Marc Métivier, Claude Lattaud

LIAP5 - UFR de mathématiques et d'informatique
Centre Universitaire des Saints-Pères
Université René Descartes (Paris 5)
45, rue des Saints-Pères
75006 Paris
{metivier, lattaud}@math-info.univ-paris5.fr

Abstract In this paper, we study the means of developing an imitation process allowing to improve learning in the framework of learning classifier systems. We present three different approaches in the way a behavior observed may be taken into account through a guidance interaction: two approaches using a model of this behavior, and one without modelling. Those approaches are evaluated and compared in different environments when they are applied to three major classifier systems : ZCS, XCS and ACS. Results are analyzed and discussed. They highlight the importance of using a model of the observed behavior to enable an efficient imitation. Moreover, they show the advantages of taking this model into account by a specialized internal action. Finally, they bring new results of comparison between ZCS, XCS and ACS.

Key words Imitation, Learning Classifier System, ACS, XCS, ZCS

1 Introduction

Reinforcement learning (RL) considers an agent that learns to perform a task by interacting with an unknown environment returning some feedback in terms of numerical reward. The agent's objective is to develop an action selection strategy which maximizes the incoming reward. A major difficulty in this learning comes from the fact that the quality of a solution relies heavily on the exploration the agent made of the environment. In fact, the only way to ensure the convergence of learning to an optimal, or even *correct*, solution is to explore repeatedly and exhaustively every state-action pairs [27]. In most cases, however, such an exploration is simply impossi-

ble. A mean of reducing the learning difficulty may consist in directing its exploration by observing the behavior of another agent. This type of learning can be obtained by an explicit teaching or a demonstration [18,31], by sharing privileged information [19], or by what is considered traditionally as imitation [1,2]. In imitation, an agent observes another agent executing an activity in context and tries to reproduce the behavior of the agent observed. Typically, the imitator agent has to express its observations of the other agents' behaviors in terms of its own capacities and to solve all ambiguities in the observations due to partial observability or noise.

The present research addresses the problem of developing imitation learning methods appropriate to reinforcement learning and especially for learning classifier systems. The principle is to enable an *observer* agent to get advantage of the observation of a *mentor* agent in order to improve its capacities to respond to its own objectives. In reinforcement learning, imitation may be considered as a knowledge transfer process between two agents [22]. For instance, the behavioral cloning approach [24,28], and, to some extent the paradigm of true imitation [20], are based on the idea of a policy transfer from one agent and the other. Utgoff's rational constraint technique [29] and Šuc and Bratko's LQ controller induction methods [26] represent an attempt to define imitation in terms of the transfer of Q-values from one agent to another. Ng's inverse reinforcement learning [21] can be seen as a model that attempts to implicitly transfer a reward function between agents. Finally, Price's implicit imitation method consists in transferring the information of the transitions model between agents [23].

The existing approaches are generally based on various assumptions. It is often admitted that a *cooperative* mentor wishes to provide inputs appropriately encoded to the agent. It is often supposed that the observer and the mentor have the same objectives or that behaviors can be represented as indivisible units that can be evaluated and then transfered. However, we consider that, in lots of cases it may be useful to remove those assumptions. For instance, a mentor may not wish, or may not be able, to alter its behavior to teach the observer. Moreover, the mentor may not want or not be able to explicitly communicate with the observer. The observation of such a mentor may nevertheless bring some useful information for the learning of the observer. For this reason, an essential point in our work is to have no assumption on the mentor's behavior. He does not have to act as a teacher. Especially, the current hypothesis according to which the mentor and observer have to share some common objectives is suppressed. Thus, in this work, the observer does not have to imitate the observed behavior. On the contrary, the point in this learning process is to enable the observer to reproduce the parts of the mentor's behavior relevant to its own objectives but also to improve this behavior. This context of work is similar to the one presented by Price [23] in his implicit imitation model applied to model-based reinforcement learning methods. One of the approaches we present in this paper is inspired from this model.

A second important aspect of the work presented in this paper is the use of learning classifier systems (LCSs) as reinforcement learning methods. It consists in rule-based systems initially developed in the research field of evolutionary computation. In those systems, perceptions of the environment are associated with actions under the form of condition-action type rules, called classifiers, using mechanisms enabling to generalize across regularities of perceptions. Their general concept lies in the generation and the manipulation of rules induced by the interactions with the environment, and the use of different mechanisms, often stochastic, to explore the set of classifiers. In particular, the classic search method consists in considering the set of rules as a population of individuals and in applying a genetic algorithm [10] on this population.

Actually, no study seems to have joined imitation learning and reinforcement learning by LCSs based on actual architectures. In 1994, Dorigo and Colombetti [9] has investigated the use of "shaping" to help a robot controlled by a LCS to learn to perform a predefined behavior. Neverthless, this work was concentrated on LCSs' architectures based on the Holland's original one [13] which is rather different from the architectures of LCS studied in the present paper. Moreover, the idea of shaping being to transfer task level knowledge by the use of a trainer, their framework is rather different from ours. LCSs being considered as model-free reinforcement learning methods, our framework is also different from the one used by Price. Consequently, this study is a first step in the design of some imitation learning methods for actual LCSs. First, we decided to consider the way an observed behavior may influence the learning of LCSs. Several approaches are presented here which are applicable to the majority of LCSs with a mentor/observer model of interaction based on explicit guidance and control taking. These approaches are tested on three major LCSs : ZCS [32] as strength-based system, XCS [33] as accuracy-based system and ACS [25] as anticipation-based system. The study presented in this paper, thus enabled some new comparisons between those systems and brings new insights into their behavior in various types of problems.

This paper begins with a general presentation of learning classifier systems, followed by a short description of ZCS, XCS and ACS. Then, we present several approaches of imitation in the framework of LCSs. We first present the mentor/observer interaction mechanism, used in our study, and then describe three different approaches of imitation in the way the observed behavior is taken into account. The third section present several experiments enabling to test and compare our approaches in different environments. In the fourth section, the results are analyzed and discussed. Finally, the last section concludes this study and presents some future works.

2 Three reference Learning Classifier Systems

In this study, we focus on three learning classifier systems (LCSs): ZCS [32], XCS [33] and ACS [25]. All these systems share a same base architecture.

They implement an online model-free reinforcement learning algorithm, and do not make use of a message list in opposite to the first LCSs [13]. Behind these common bases, each of them adds some specific features modifying the way a problem is learned and decisions are taken.

2.1 The global framework

LCSs are essentially rule-based decision systems in which generalization is obtained through the use of evolutionary methods [10]. A LCS is always linked to an *input interface* and an *output interface* used to communicate with the environment. At any instant t, it gets from the input interface a sensory *situation* σ_t representing the perception of the actual state of the environment and a reward ρ_t. Such a situation is defined as a string of discrete values: $\sigma_t \in S^L$ with L the size of the string and $S = \{s_1, \ldots, s_m\}$ a set of m possible values. The reward ρ_t is a simple real value.

In response, the LCS selects an action α_t and sends it to the output interface which has the function to execute this action in the environment. Actions are coded with discrete values: $\alpha_t \in \{a_1, \ldots, a_n\}$, where n is the number of different actions and $a_1, \ldots, a_n$ are these actions.

2.1.1 Coding of the information The principal component of any LCS is the *classifiers population*. It represents the basic knowledge of the system and consists in a list of syntactic rules using the form $<$ *condition* $>:<$ *action* $>$. These rules, called *classifiers*, are composed of a condition part and of an action part:

- the condition part specifies the prerequisites of the perception for this rule to be applicable.
- the action part specifies an action to execute if the condition part is satisfied by the perception.

The condition part is represented by a string of L discrete values in $S \bigcup \{\#\}$. The particular "#" symbol is called *generalization* symbol or *don't-care* symbol. It matches every possible attribute value of the perception. Thus, if $S = \{0, 1\}$, a condition $C = 00\#1$ is satisfied by perceptions $\sigma_t = 0001$ or $\sigma_t = 0011$. We say that this condition *matches* these situations.

The action part can have several forms. It may be an action directly achievable in the environment. In this case, it will be one of the n possible actions $\{a_1, \ldots, a_n\}$.

In addition to those two parts, a classifier usually contains some parameters making it possible to measure its importance for the action choice made by the system. In particular, any classifier contains a measurement of the interest which its action represents in the environment when its condition is satisfied by the perception. These parameters depend on the LCS.

2.1.2 Rules management At any cycle t, the system perceives a couple (σ_t, ρ_t) from the environment, where σ_t is the current situation, i.e. the perception of the current state of the environment, and ρ_t a reward value associated to that state. As in any decision system, the LCS has to determine an action value $P(a_i)$ for every action a_i and to choose an action according to those values. The action value function computed by the LCS is often called *prediction function* because it generally consists in predicting a discounted reward as the same of the Q function in model-free reinforcement learning methods such as Q-learning [30]. In order to compute those values for the current cycle, the LCS creates a subset of the classifiers population containing every classifier whose condition part matches the current situation. This set is called *match set*. The action values are then calculated according to the parameters of the classifiers in this set. Every classifier contributes to the computation of the value of the action advocated by its action part.

Once computed the action values $P(a_i)$, the action choice is made by a selection mechanism enabling to favor the exploration of the environment or the exploitation of the system's knowledge. The most common mechanism is probably ϵ-greedy [27] in which, with a probability ϵ, the action is chosen randomly in the set of possible actions, otherwise the action with the highest value is chosen.

The selected action is then executed in the environment and the system perceives in consequence a new couple $(\sigma_{t+1}, \rho_{t+1})$. This information allows to update the classifiers population. This update consists in two steps:

- the reinforcement: the parameters of the classifiers contained in the match set and advocating the executed action are updated. The update mechanisms depend on the considered LCS type. Most of time, they apply the bucket brigade algorithm [10], or Q-learning update equation [30].
- the discovery: some new classifiers are created and some classifiers may be eliminated. Several methods are used according to systems. The original method consists in using a genetic algorithm (GA) [10]. In some systems, a *covering* operation is added whose function is to create new classifiers if the match set is empty. Finally, some LCS use specific heuristic instead of a GA.

Finally, the $t+1$ cycle can begin with $(\sigma_{t+1}, \rho_{t+1})$ as current perception.

2.2 ZCS

ZCS has been presented by Wilson in 1994 [32]. It is considered as the *zeroth level* classifier system, i.e. a LCS which conserves only the bare minimum of classical LCS. Indeed, ZCS uses a classifier format having a unique parameter called *strength* and its condition and action parts are expressed in the binary alphabet $S = \{0, 1\}$. The classifiers update is made according to the

implicit bucket brigade algorithm [10]. A kind of "strength conservation" is applied. As a matter of fact, the classifier has to pay a tax to participate to the action choice. The sum of those taxes is distributed among classifiers which, at the last cycle, advocated the last executed action. When a new action is selected and executed in the environment, the perceived reward is distributed among the classifiers of the match set advocating this action. Finally, the discovery of some new classifiers is made by using a genetic algorithm.

Action values are computed, for each action a, as the sum of the strengths of the classifiers contained in the match set and advocating a:

$$P(\sigma_t, a) = \frac{\sum_{c \in M_t|a} force(c)}{\sum_{c \in M_t} force(c)} \quad (1)$$

with σ_t the current situation, M_t the match set, $M_t|a$ the set of classifiers in M_t which advocate the action a, and $force(c)$ the projection function which associates to a classifier c its strength value.

The first experiments made by Wilson showed that ZCS was able to solve some maze problems but, often, with suboptimal strategies. Cliff and Ross [7] highlighted ZCS incapacity to solve some multi-steps tasks implying long action chains. Nevertheless, Bull and Hurst showed, in 2002, that ZCS was able to obtain optimal performances, even in long chain environments, with appropriate parameter values [3].

2.3 XCS

XCS has been presented by Wilson in 1995 [33] and is considered as the reference of accuracy-based LCS. It makes use of a genetic algorithm in which classifier's fitness is a measure of the accuracy of the classifier reward prediction instead of their strength. Thus, in XCS, the strength of classifiers is replaced by three parameters: the *reward prediction*, the *prediction error* and the *fitness*. The *reward prediction* estimates the reinforcement perceived form the environment when the classifier action is executed. This parameter is used during the action selection. The *prediction error* is computed by comparison between the prediction and the real reward. Finally, the *fitness* measures the prediction accuracy: it is computed in function of the reverse of the error. It is used for the action selection and the genetic algorithm selection. XCS classifiers include several other additional parameters less fundamental than the three precedents. Those parameters are used to accelerate the learning process and to control more precisely the call of the discovery mechanisms. Like ZCS, the condition parts and action of its classifiers use a binary alphabet.

The value of each action is calculated as the average of the prediction of classifiers in M_t advocating the considered action, balanced by their fitness.

$$P(\sigma_t, a) = \frac{\sum_{c \in M_t|a} pred(c) \times fit(c)}{\sum_{c \in M_t|a} fit(c)} \quad (2)$$

with σ_t the current situation and $M_t|a$ the set of classifiers in M_t which advocate the action a. $pred(c)$ and $fit(c)$ are two projection functions which, to a classifier, associate respectively its prediction value and its fitness value.

The classifiers predictions update is inspired from Q-learning. Thus, the prediction of classifiers contained in the match set M_t and advocating the last executed action α_t, is updated according to the following equation:

$$p \rightarrow (1-\beta) \cdot p + \beta \cdot (\rho_{t+1} + \gamma \cdot \max_{a \in A}\{P(\sigma_{t+1}, a)\}) \quad (3)$$

with p the updated prediction parameter, β the learning rate, γ the discount factor, and A the set of possible actions. ρ_{t+1} is the reward perceived in consequence of α_t execution.

XCS particularity is to favor classifiers with a correct prediction of the discounted reward instead of those with a high prediction. This enables to develop a population of classifiers modeling accurately the distribution of the reward in the environment. In 2000, Lanzi presented a formalization of LCS in a reinforcement learning perspective [17]. He demonstrated that XCS is formally equivalent to tabular Q-learning if generalization and the genetic algorithm are turned off, and if the population size is not limited. With generalization and the genetic algorithm, the classifiers population of XCS evolves so as to approximate the action value function as it is defined by Q-learning.

XCS has been the subject of many studies. In 1996, Kovacs showed that this system has the capacity to develop a minimal representation of the optimal solution [14]. The relation between the performances of XCS and the complexity of the problems was approached by Kovacs and Kerber [15]. Lanzi showed that XCS presented serious difficulties of modelling in certain types of environment, where certain states are not explored *sufficiently often* [16].

2.4 ACS

ACS has been presented by Stolzmann in 1997 [25]. Its particularity is to apply a learning mechanism issued from the field of psychology: the "anticipatory behavioral control" of Hoffmann [12]. In this LCS, the basic structure of classifiers is enhanced with an *effect part* representing an anticipation of the perceptive consequences of actions in the environment. This effect part has the same structure of the condition part except that the # symbol does not have the same sense as it means that the value of the attribute containing this symbol is not changed by the execution of the action advocated by the classifier. The classifiers of ACS have two parameters: the reward prediction and the anticipation quality. The effect part, which is associated to a learning process called *anticipatory latent learning* (ALP), allows the system to learn latently an internal representation states dynamic of the environment. Thus, the classifiers population evolves so as to model transitions of the environment.

The value of an action is computed as the maximum product (prediction × quality) of classifiers in M_t which advocate this action:

$$P(\sigma_t, a) = \max_{c \in M_t|a} (prediction(c) \times quality(c)) \quad (4)$$

with σ_t the current situation and $M_t|a$ the set of classifiers in M_t which advocate the action a. $prediction(c)$ and $quality(c)$ are two projection functions which, to a classifier c, associates respectively its prediction value and its quality value.

As in XCS, the predictions update are inspired from Q-learning (see equation 3). The quality parameter is updated during the ALP according to the following principle: if the updated classifier correctly anticipates the situation perceived in consequence of the execution of the action, its quality is increased, or decreased, according to the Widrow-Hoff delta rule:

$$\text{Increase} : q \leftarrow (1 - b_q) \cdot q + b_q \quad (5)$$

$$\text{Decrease} : q \leftarrow (1 - b_q) \cdot q \quad (6)$$

with q the updated quality parameter and b_q the quality learning rate.

In ACS, the discovery is not obtained with a genetic algorithm. The classifiers population begins with one completely general classifier for each possible action: every attributes of the condition and effect parts are # symbols. Then the ALP creates new specialized classifiers as soon as some error of anticipation are detected.

The ACS model has been the subject of many evolutions and studies over the last few years. The version of ACS used in this article is based on that of Stolzmann and includes various improvements made by Stolzmann and Butz. It acts as the version presented in [5] before the addition of the genetic algorithm investigated in the article.

3 Approaches of imitation in LCSs

The objective of this work is to equip known systems with some mechanisms, making it possible to imitate implicitly certain elements of the behavior of a mentor. This work was carried out in several stages. Initially, we defined a context of imitation and of mentor/observer interaction as a base study. Then several approaches in the way an observed behavior may be taken into account are presented.

3.1 A study framework

3.1.1 The learning agent. We consider an agent, said *observer*, interacting with an environment according to the classical framework of reinforcement learning, and whose control architecture is a classifier system. Thus, at

each execution cycle t, the agent perceives a couple $(\sigma_t,\ \rho_t)$ which represents respectively the current situation of the environment and a reward corresponding to this situation. Its control architecture selects an action α_t according to this couple. σ_t is defined as a string of L symbols among the set $S = \{s_1, \ldots, s_m\}$. The reward is a positive or null real value. Finally, α_t is an element of a set of n actions $A = \{a_1, \ldots, a_n\}$. Once selected, the action is executed in the environment. A new cycle begins with the perception of the situation σ_{t+1} and the reward ρ_{t+1} as consequences of the execution of this action.

3.1.2 The mentor. In parallel to the observer agent, we consider a mentor which can interact in the same environment with a given policy. Whatever the mentor's properties, its behavior is regarded as containing information likely to help the observer maximize the cumulative reward of the environment. To take into account this behavior, it is necessary to define a *model of interaction mentor/observer* to specify how this behavior will be transmitted to the observer.

The model of interaction used in this study is called *guidance interaction*. This method consists in authorizing the takeover of the observer by an external agent, either a human being or a machine. Thus the mentor does not appear explicitly in the environment in the shape of a second agent. It acts in the environment via the observer. The guidance interaction consists of four rules:

- At any moment during a simulation, a *user* (or *mentor*) can take the control of the observer agent.
- Any mentor controlling the system perceives in the environment the situation from the sensors of the system and only this situation.
- Any mentor controlling the system must choose an action, at each execution cycle, among the actions set of the system. The chosen action replaces the action selected by the system (mechanism of subsumption) and is returned to the actuators of the system.
- At any moment during a simulation, the mentor guiding the observer can stop the takeover. The observer agent then retrieves its autonomy.

Figure 1 illustrates the operation of guidance. The situation σ_t is sent to the mentor in parallel of the control architecture. The action selection is assisted by a mechanism of subsumption making it possible for the action of the mentor to replace the action chosen by the control architecture in the case of guidance. If this mechanism receives an action from a mentor, then it is this action which will be sent to the actuators. Formally, this mechanism is represented by the *subaction* operator applied to the action returned by the action selection mechanism of the observer and to the mentor's action. We define this operation as follows:

$$subaction(a_o, a_m) = \begin{cases} a_o & \text{if } a_m = \emptyset, \\ a_m & \text{otherwise.} \end{cases}$$

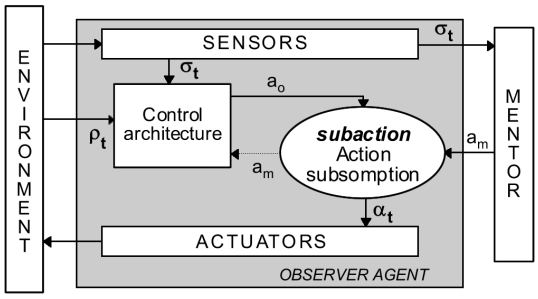

Fig. 1 Guidance interaction mechanism.

with a_o, a_m two actions. During subsumption, a_o will be the action chosen by the control architecture and a_m the action of the mentor. Thus, in the case of guidance ($a_m >\!\!\neq \emptyset$), the mentor replaces the action selection mechanism of the system. The action finally sent to the actuators is: $\alpha_t = subaction(a_o, a_m)$.

This way of transmitting a behavior seems as an elementary method, and finds its application in several problems. In particular, the takeover is a frequent mechanism in the fields of robotics or Virtual Worlds [11]. Let us note that, within the framework of the Virtual Worlds, this process is also called *avatarization*: during the guidance, the observer is an avatar [8] of the mentor, i.e. his representation in the environment. The behavior transmission method by guidance remains however general. Its characteristic, besides facilitating the expression of the mentor behavior, is to impose systematically the test by the agent of the actions of the mentor. This systematic test is important for model-free methods of reinforcement learning since, not taking advantage of any transition model of the environment, it is the only way by which those methods can determine the consequences of an action.

3.1.3 Imitation approaches Our approaches of imitation have to allow the observer agent to improve his behavior thanks to the observation of the mentor's behavior. Considering the mentor/observer interaction model used, the observer will take account of the behavior of the mentor when it is guided and will then have to hold the parts of this behavior which can improve its own performances.

We present three different approaches we developed to explore the way a behavior observed may be taken into account. They distinguish themselves by the way they integrate this behavior. The first consists in letting guidance act like a simple method of exploration of the environment, while the two others make use of a model of the mentor built during guidance, i.e. a system whose purpose is to give a value to every actions according to the behavior of the mentor. The difference between those two last approaches is in the

type of values given to the actions of the mentor and in the way those values are taken into account for the action selection of the observer agent.

3.2 "Guidance-only" approach

The *guidance-only* approach (GO) consists in letting guidance replace the action choice mechanism of the LCS to which it is applied. Each time a mentor will impose an action on the system, this one "will believe" to have chosen this action and will learn according to this action. Because the mentor replaces the action choice mechanism, the guidance mechanism then becomes an action choice mechanism as well as ϵ-*greedy* [27]. It acts as a method of exploration having to do with doing what somebody else says. Thus, the mentor guides the exploration of the system.

The goal of this approach is to allow the study of guidance as a method of exploration in the learning carried out by LCSs. This approach does not bring any particular mechanism to support the imitation. It was implemented, on the one hand, in order to highlight the impact of guidance on LCSs and, on the other hand, to obtain performances to use as a bases in the study of the other approaches.

3.3 "Augmented Bellman equation" approach

The *augmented Bellman equation* approach (ABE) adapts the implicit imitation model of Bob Price[23] in the context of LCSs. It makes use of a LCS dedicated to the modelling of the reward specific to the mentor's behavior in parallel of the LCS to which it applies and *amalgamates* their predictions on a similar basis than the augmented Bellman equation presented by Price.

3.3.1 Price's implicit imitation The augmented Bellman equation has been presented by Price in order to implement a model of imitation in the context of model-based reinforcement learning in Markov decision processes. As the classical Bellman equation, it defines the state value function of the learning agent (V) from the estimated transition function (T), but it uses in addition a second transition function computed according to the observation of a mentor. This additional transition function is a Markov chain T_m, defined on S, resulting of the mentor's behavior, with $T_m(s,t)$ denoting the transition probability from the state s to the state t.

The augmented Bellman equation reads as follows:

$$V(s) = R(s) + \gamma \cdot \max\Big\{\max_{a\in A}\Big\{\sum_{t\in S} T(s,a,t)V(t)\Big\}, \sum_{t\in S} T_m(s,t)V(t)\Big\} \quad (7)$$

Thus, it consists in the usual Bellman equation with an additional term $\sum_{t\in S} T_m(s,t)V(t)$, called the "second sum", and representing the value expected of the reproduction of mentor's action. The estimated values produced by this equation are called *augmented values*.

The ABE approach is inspired by this method and adapts this equation to the framework of LCSs. It implements the notion of *augmented prediction*.

3.3.2 From augmented values to augmented predictions LCSs may be considered as model-free methods of reinforcement learning: they do not dispose of a transition model of the environment and compute state-action values rather than state values. Those action values are directly calculated and updated during the interaction with the environment. Their updating is done in order to tend to the optimality Bellman equation for Q:

$$Q(s,a) = \rho + \gamma \cdot \max_{a' \in A} \Big\{ Q(s',a') \Big\} \tag{8}$$

with $Q(s,a)$ the value of the action a in the situation s. A is the set of possible actions and, s' and ρ are respectively the situation and the reward resulting from the execution of a.

We call $P_o(s,a)$ the action value function of the observer agent, computed by its control architecture. Adapting the augmented Bellman equation in the context of LCSs consists then in using a second action value function proper to mentor's behavior and to integrate it in the computation of P_o by maximization. We thus define P_m as an action value only computed according to observed actions of the mentor. The P_o and P_m functions are define as follows:

$$P_m(s,a) = \rho + \gamma \cdot \max_{a' \in A} \Big\{ P_m(s',a') \Big\} \tag{9}$$

$$P_o(s,a) = \rho + \gamma \cdot \max_{a' \in A} \Big\{ P_o(s',a'), P_m(s',a') \Big\} \tag{10}$$

where s is a situation in S and a an action of the actions set A. s' and ρ are respectively the situation and the reward resulting from the execution of a. We call *augmented prediction* the values represented by the P_o function and *guided prediction* those represented by P_m.

3.3.3 ABE Architecture Figure 2 presents the architecture of the *augmented Bellman equation* approach (ABE). This architecture applies the principle of augmented predictions by the using of two LCSs in parallel:

- RwdCS ("Reward Classifier System") implementing the computation of the augmented predictions P_o of the system.
- UsrRCS ("User Reward Classifier System") whose aim is to compute the guided predictions P_m.

At each execution cycle t, the system gets the couple (σ_t, ρ_t) representing its perception of the current state of the environment and the reward corresponding to this state. According to σ_t, the two LCSs compute their respective predictions $P_o(\sigma_t, a)$ and $P_m(\sigma_t, a)$ for each possible action a. These computations are made using the predictions of their classifiers and their internal mechanisms. They depend on the types of LCSs used as RwdCS

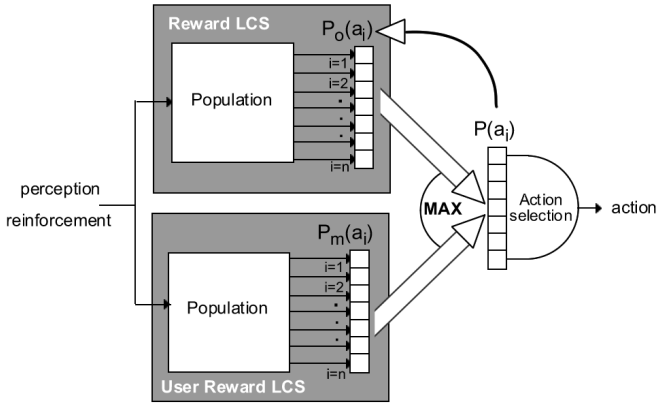

Fig. 2 The ABE architecture.

and UsrRCS. The set of actions used by these two LCSs is A, i.e. the actions set of the observer.

In the ABE architecture, the augmented Bellman equation is applied so that the selection of the action uses the augmented predictions. Instead of using equation 10 directly at the time of predictions updating, it is applied implicitly by a maximization of the predictions made by the two LCSs during the action choice. Thus, at the end of the predictions computation, the system has two predictions $P_o(\sigma_t, a_i)$ and $P_m(\sigma_t, a_i)$ for each action a_i. These predictions are maximized for each action in order to obtain a set of global predictions $P(\sigma_t, a_i)$:

$$P(\sigma_t, a_i) = \max\Big\{P_o(\sigma_t, a_i), P_m(\sigma_t, a_i)\Big\} \tag{11}$$

The predictions represented by the P function are those used by the action choice mechanism. An action α_t is then selected. Please note that this action may be chosen by a mentor in the case where the system is in a guidance period. In system's perspective, we can, however, consider guidance as an action choice mechanism as others. The selected action is then executed in the environment and the system receives an immediate reward ρ_{t+1} and a new perception σ_{t+1}.

That new information allows to update the classifiers of RwdCS and UsrRCS. But UsrRCS's update is only done in the case of guidance. Thus, this LCS predicts the environmental reward according to the mentor's behavior exclusively, i.e. the reward obtained when the mentor acts. RwdCS, however, is updated at each execution cycle and uses for it the maximum of P_o and P_m predictions in order to get augmented predictions. To obtain this, global prediction values P comes to replace P_o predictions before RwdCS update:

$$P_o(\sigma_t, a) \leftarrow P(\sigma_t, a) \tag{12}$$

for each action a of A. Thus, the update of classifiers of RwdCS is made by using predictions maximizing those issued from RwdCS and those issued from UsrRCS.

3.3.4 UsrRCS: a mentor's reward model In our study, we say that ABE "is applied to" a certain kind of LCS to signify that RwdCS will be of that type. For UsrRCS, a minimalist classifier system has been developed in order to make the global study of this approach easier.

UsrRCS uses the classifier structure $(c : a \rightarrow p)$ with c the condition part, a the action part and p a reward prediction in $\mathbb{R}$. To facilitate the study of the impact of this approach, we do not use generalization in this LCS. Thus the condition part of UsrRCS does not allow the *don't care* symbol in the values set of its elements.

The population of UsrRCS is not limited in size. UsrRCS begins with an empty population and creates classifiers by a covering method: at each execution cycle t, if the match set is empty, then a new classifier (σ_t, a_i, p_0) is created for each possible action a_i of A, with σ_t the current perception and p_0 a constant parameter used to initialize predictions. This mechanism is made so that each couple (perception, action) will be represented by only one classifier of the population. Thus, any match set will contain just one classifier for each distinct action. For this reason, action values are very simple to compute: for each action, it consists in the prediction value of the unique classifier of the match set which advocates this action. After each execution of an action, in the case of guidance exclusively, the prediction of the classifier having advocated this action is updated as follows:

$$p \leftarrow (1 - \beta) \cdot p + \beta \cdot (\rho_{t+1} + \gamma \cdot \max_{a \in A} \{P_m(\sigma_{t+1}, a)\}) \tag{13}$$

where β is the learning rate and γ is the discount factor of UsrRCS. A is the set of possible actions.

3.4 "Imitation action" approach

The *imitation action* approach (IA) uses a behavioral model of the mentor. Unlike the ABE approach, this mentor model is not used in parallel of the base LCS of the agent. In this approach, the mentor's modeling does not take into account the rewards perceived from the environment. In addition to the classical actions, the main LCS *chooses* to follow the behavior of the mentor, or not, through a specific action which is called the *imitation action*.

3.4.1 IA architecture The figure 3 presents the architecture of the *imitation action* approach (IA). This approach uses two specialized LCSs:

- RwdCS ("Reward Classifier System") implements the computation of global predictions of the system. Thus, it plays the role of the main LCS

as the RwdCS in the ABE approach. Unlike the ABE approach, however, the set of actions used by RwdCS in the IA approach is $A \cup \{\alpha_{im}\}$, with α_{im} an additional action called *imitation action*.
– UsrBCS ("User Behavior Classifier System") models the behavior of the mentor. Its aim is to predict actions of the mentor whatever the rewards of the environment. The set of actions used by UsrBCS is A.

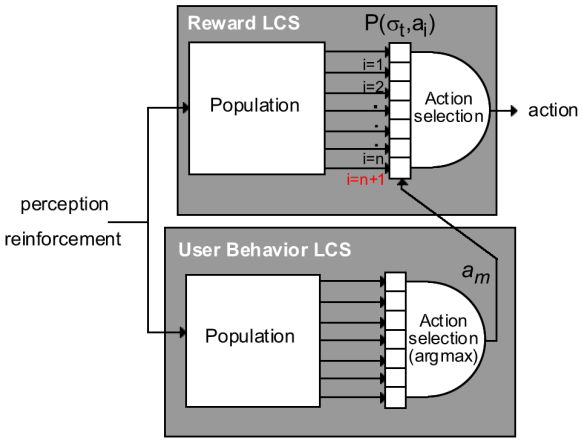

Fig. 3 The IA architecture.

In this architecture, the imitation action α_{im} is an internal action. Unlike other actions, also called *external* actions, it is not executable in the environment. The function of α_{im} is to act on the way the system will select an external action to send to the actuators.

At each execution cycle t, the system receives the couple (σ_t, ρ_t) representing its perception of the current state and the reward corresponding to this state. According to σ_t, RwdCS computes its prediction values $P(\sigma_t, a)$ for each action a in $A \cup \{\alpha_{im}\}$. Those calculations are effected using classifier's predictions and the internal mechanisms of RwdCS. They depend on the type of LCS used as RwdCS.

The predictions $P(\sigma_t, a)$ are used to select an action in $A \cup \{\alpha_{im}\}$. If an action of A is selected, then this action is sent to the actuators. If α_{im} is chosen, the action selection mechanism requires UsrBCS to obtain the action in A which is considered similar to the mentor's behavior. It is finally this last action which is sent to the actuators. The external action is then executed in the environment and the system perceives as a result an immediate reward ρ_{t+1} and a new situation σ_{t+1}.

This new information allows to update the classifiers of RwdCS and UsrBCS according to their respective internal mechanisms. Note that the imitation action is an action like any other in the functioning of RwdCS. If

this action is selected, then the immediate reward and the next perception are considered by RwdCS as the consequences of this action, and not of the external action finally sent to actuators. Thus, as any other action, α_{im} will be associated to a reward in RwdCS. In the case of guidance by a mentor, RwdCS considers that the action executed is always α_{im}.

As UsrRCS in the ABE approach, the classifiers of UsrBCS are only updated during cycles where a mentor guides the observer agent. The distinctive feature of UsrBCS is that its updates are not made according to the rewards nor the next perception. Indeed, its objective being to predict mentor's actions, its predictions are just updated according to the current perception and the action chosen by the mentor during guidance.

3.4.2 UsrBCS: a mentor's behavioral model As for UsrRCS, a minimalist LCS has been used to act as UsrBCS. It uses the classifier structure ($c : a \rightarrow d$) with c the condition part, a the action part and d a prediction of the correspondence degree between this classifier and the mentor's behavior. d is defined as a real value in $[0, 1]$. As in UsrRCS, the generalization is not used in order to facilitate the study of the approach.

The population of UsrBCS is not limited in size. UsrBCS begins with an empty population and creates classifiers by the same covering method than in UsrRCS with the parameter d_0 as constant parameter used to initialize the degree predictions of new classifiers. At each execution cycle, the value of each action is defined as the correspondence degree of the unique classifier in the match set that advocates this action. Finally, during the guidance, each classifier ($c : a$, d) in the match set is updated as follows:

$$d \leftarrow \begin{cases} (1-\beta)\cdot d + \beta & \text{if } a = \alpha_t, \\ (1-\beta)\cdot d & \text{otherwise.} \end{cases}$$

where β is the learning rate, and α_t is the action executed in the environment.

4 Evaluation

4.1 Experimental conditions

4.1.1 Environments The experiment context used to evaluate our approaches is the one defined by Wilson in his woods environments. It consists in 2D discrete environments in which each cell may either be empty or contain an obstacle or some *food*. At each cycle of simulation, the agent perceives the eight cells close to its position and chooses an action among the eight actions consisting in moving on one of those close cells. Three environments of the woods type are used. They have been selected so as to propose a problem of increasing complexity to our agent.

4.1.2 Organization of experiments A simulation is defined as a succession of problems. A problem begins with the setting of our agent in an empty cell chosen randomly. Then, the agent acts in the environment until it reaches a food cell or until 500 actions have been executed. The problem is then considered as finished and a new problem may begin. In the case where a food cell is reached, the agent gets a reward whose value is 1000. Three types of problems can be distinguished:

- exploration problems: the agent selects its actions with an exploration mechanism. This mechanism depends on the type of LCS considered.
- exploitation problems: the agent applies pure exploitation. It always selects the action corresponding to the highest prediction. This type of problem is used to compute the performances of the system.
- guidance problems: the agent is controlled by a mentor. The behavior of the mentor depends on the considered experiment.

The simulations are composed of several phases: guidance, exploration and exploitation. Each of them is constituted of problems of the same nature. In order to evaluate the system during all the phases, however, exploration and guidance problems alternate with exploitation problems. The performances of the system are computed during each exploitation problem of the simulation. For each experiment, the number of problems in each phases is chosen so as to let the time for performances to converge.

Two performance measures are achieved in order to evaluate our approaches:

- the *number of steps* to reach the food cell which is the number of actions executed in the environment during a problem.
- the *imitation percent* of the mentor: it consists in the percent of actions executed by the agent which corresponds to the behavior of the mentor during a problem.

The performances results presented later in this paper are mean value of all the performances obtained in ten simulations as soon as the performances are stabilized. More precisely, ten simulations are effected and curves of performances are obtained. Then, a problem is selected from which all curves had become stable. Finally, the performance value presented is a mean of all the values of the curves obtained after the chosen problem. In the case of the *number of steps* performances, we also give the standard deviation of the values used in the mean computation. Finally, some graphs showing the dynamics of the steps to food performances are given in complement of the global means.

4.1.3 Application to ZCS In order to apply the ABE approach on ZCS, it is necessary to modify the method used by ZCS to compute its action value. Indeed, the action values are normalized in ZCS and their computation is based on the reward prediction. To enable the application of EBA approach, we suppressed the normalization. Thus, the action value of an action a_i is

simply defined as the sum of the strength of the classifiers in the match set which advocate this a_i. From then on, it is no more divided by the sum of the strength of all classifiers in the match set. The obtained equation is the following:

$$P(a_i) = \sum_{c \in M|a} strength(c) \tag{14}$$

with M the match set, $M|a$ the subset of M containing the classifiers advocating the action a, and $strengh(c)$ the projection function defined on the set of classifiers which associates to a classifier c its strength parameter. Please note that this modification doesn't affect the system functioning since the modified predictions are only used in the action selection mechanism which selects actions in proportion to their predictions.

Another specificity of ZCS is to have been designed to use a binary alphabet for the condition and action parts of classifiers. The set of symbols used is {0,1} to which is added the *don't-care* symbol # used for generalization. The contents of the perceived cells are encoded as follows: 00 for an empty cell, 10 for an obstacle cell, and 11 for a food cell. The agent perceived the eight cells surrounding its position from north and turns clockwise. The actions are numbered from 0 to 7 and are encoded with three binary symbols. The action numbered 0 is the one associated to the movement to the north cell and the others actions are obtained turning clockwise. In the IA approach, the actions are coded with four binary symbols: external actions are numbered in the same way with three symbols and the imitation action is represented by the string 1000.

Finally, the values of the parameters of ZCS used in our experiments are the following: $N = 400$, $S_0 = 20$, $\beta = 0,2$, $\gamma = 0,71$, $\tau = 0,1$, $\Phi = 0,5$, $P_\# = 0,33$, $\chi = 0,5$, $\mu = 0,002$, $\rho = 0,125$. During exploration problems, the actions are chosen with the roulette wheel method, the selection values being the predictions. The parameters values, the exploration mechanism and the encoding of perceptions and actions are the ones used by Wilson to evaluate ZCS' performances in [32]. During exploitation problems, the genetic algorithm is disabled.

4.1.4 Application to XCS As ZCS, XCS has been designed to use a binary alphabet to encode condition and action parts of the classifiers. In our experiments, it uses the same encoding that ZCS presented in the last section.

The values of the parameters of XCS we used follows: $N = 800$, $\beta = 0,2$, $\gamma = 0,71$, $\chi = 0,5$, $\mu = 0,05$, $\Theta_{mna} = 8$, $P_\# = 0,33$, $P_I = 0,01$, $err_I = 0$, $F_I = 0,01$, $\alpha = 0,1$, $\epsilon_0 = 10$, $\mu = 5$, $\Theta_{GA} = 30$, $\Theta_{del} = 20$, $\delta = 0,1$. It consists in default values chosen according to the parametrization advices given by Butz in its algorithmic description of XCS [6]. In the case where the IA approach is applied to XCS, the θ_{mna} parameter is set to 9 to take account of the imitation action. During exploration problems, the actions are chosen using the ϵ-greedy method with $\epsilon = 0.8$. GA-subsumption is

on with $\theta_{sub} = 20$, while action-set subsumption is off. Finally, the genetic algorithm is disabled during the exploitation problems.

4.1.5 Application to ACS Unlike ZCS and XCS, ACS is not based on a binary alphabet. Thus, empty cells are represented by 0, obstacle cells by 1 and food cells by 2. The actions are numbered from 0 to 7 from the one associated to the movement to the north and turning clockwise. The imitation action of the IA approach is numbered 8.

Finally, the parameters of ACS have the following values in our experiments: $\gamma = 0,71$, $b_r = 0,2$, $b_q = 0,2$, $\theta_r = 0,9$, $\theta_i = 0$. In exploration problems, the action is chosen with the following mechanism: with a probability $p_x = 0.8$, the action is selected randomly, otherwise the action is selected by roulette wheel with the action values used as selective values. This mechanism has been used by Butz in [4].

4.1.6 Settings of the mentor modeling LCSs When the ABE approach is applied, the parameters of UsrRCS have the following values: $\gamma = 0,71$, $\beta = 0,2$, $p_0 = 0,01$. When the IA approach is applied, the parameters of UsrBCS have the following values: $\beta = 0,2$ et $d_0 = 0,01$.

4.2 Experiments in Woods1

Woods1 is a Markov environment presented by Wilson for the evaluation of ZCS [32]. It consists in a very simple environment (figure 4) in which the mean performance of an optimal behavior is of 1.6875 steps to reach the food. Wilson has shown that ZCS develops a stable sub-optimal behavior in this environment [32]. In [33], he showed that XCS is able to reach an optimal behavior in the Woods2 environment whose structure is similar to Woods1 but containing two types of obstacles and foods. Those results let us to believe that XCS should develop an optimal policy in Woods1. Finally, Butz showed in [4] that ACS develops a complete environmental model of Woods1 and an optimal behavior to reach the food.

The experiments done in Woods1 enables to observe the effect of guidance on LCSs according to the considered approaches. Two series of experiments have been executed: the first one with an optimal mentor, and the second one with a sub-optimal mentor. The policies of those two mentors, respectively named UO and $U1$, are presented in figures 4b and c. Each experiment begins with a guidance phase of 200 problems, followed by an exploration phase of 800 problems, and finally an exploitation phase of 2000 problems. The steps to food performances are compared with the LCSs' performances without guidance. They are obtained with an experience composed of an exploration phase of 1000 problems, followed by an exploitation phase of 2000 problems. All the presented results are a mean of the performances obtained during the last 2000 problems.

The obtained results are presented in tables 1 and 2. Moreover, figures 5a and 5b present the steps to food dynamics of all the experiences including

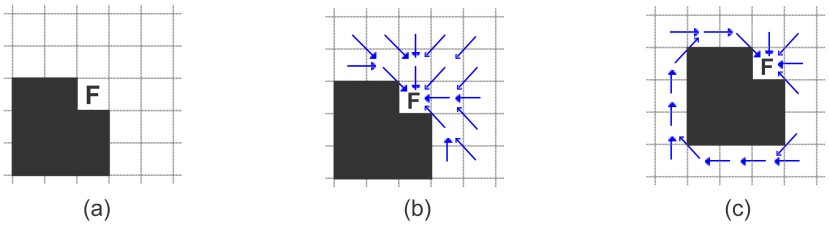

Fig. 4 (a) The Woods1 environment. It is toroid: this pattern is indefinitely repeated in the horizontal and vertical directions. (b) Behavior of the mentor $U0$. This policy is optimal. (c) Behavior of the mentor $U1$. The mean number of step of this policy is of 4.19.

	ZCS			XCS			ACS		
Approach	stf	σ_{stf}	im	stf	σ_{stf}	im	stf	σ_{stf}	im
No guidance	3.72	0.10	-	1.69	0.02	-	1.69	0.02	-
GO	2.84	0.06	55	1.69	0.02	91	1.69	0.02	85
ABE	1.74	0.02	96	1.69	0.02	92	1.69	0.02	86
IA	2.12	0.04	75	1.69	0.02	100	1.69	0.02	100

Table 1 Mean performances in Woods1 with the mentor $U0$. "stf": steps to food performance. "σ_{stf}": standard deviation of the steps to food performance. "% im": percent of action imitating the mentor's behavior.

	ZCS			XCS			ACS		
Approach	stf	σ_{stf}	im	stf	σ_{stf}	im	stf	σ_{stf}	im
No guidance	3.72	0.10	-	1.69	0.02	-	1.69	0.02	-
GO	2.78	0.05	50	1.69	0.02	63	1.69	0.02	63
ABE	2.98	0.12	84	1.69	0.02	63	1.69	0.02	63
IA	2.62	0.07	72	1.69	0.02	63	1.69	0.02	63

Table 2 Mean performances in Woods1 with the mentor $U1$. "stf": steps to food performance. "σ_{stf}": standard deviation of the steps to food performance. "% im": percent of action imitating the mentor's behavior.

ZCS. With these results, we may observe that ZCS has a mean number of steps stable but sub-optimal of 3.72. These performances are similar to those obtained by Wilson. Moreover, we can observe that passing by a guidance phase has allowed to obtain better performance than without guidance. This observation is made even with the mentor $U1$ nevertheless its policy less efficient than ZCS without guidance. In this case, the system improved its performance by reproducing some of the efficient actions of the mentor especially in the cells neighboring the food. Thus, it had implicitly imitated the efficient parts of the mentor's behavior and dropped the useless parts. When the mentor is $U0$, then the performance obtained with the

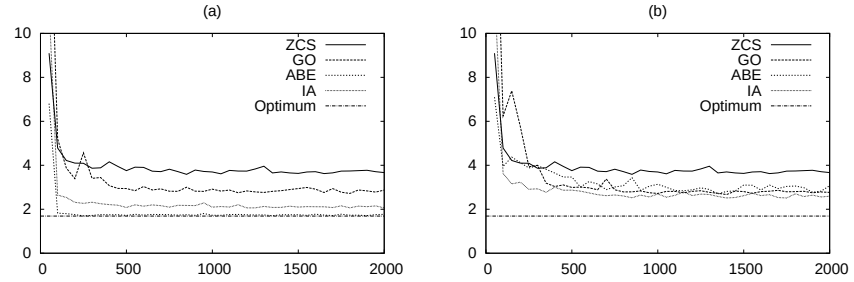

Fig. 5 Steps to food performances with ZCS in Woods1. (a) The mentor is $U0$. (b) The mentor is $U1$.

ABE approach becomes optimal, and the one of the IA approach close to optimal. This is confirmed by the fact that the system follows the mentor's behavior in 96% of its actions with the ABE approach and in 75% with the AI approach.

XCS and ACS develop optimal policies without guidance. This is in accordance with results published by Wilson and Butz. With a guidance phase and the use of our approaches, developed behavior are optimal too. Nevertheless, these results enable to make some observation about the way the mentor's behavior can act upon the learning. Indeed, we may notice that the imitation percent of the mentor $U0$ tends to 100% with the IA approach. With this approach, the system develops a policy exactly as the same as $U0$'s policy whereas the other approaches reproduce this behavior only partially: about 91% for XCS and 85% for ACS.

4.3 Experiments in Woods14

Woods14 is a Markov environment consisting in a linear path of 18 empty cells (figure 6) leading to a food cell. The mean number of steps of an optimal behavior in this environment is of 9.5. This environment has been introduced by Cliff and Ross in order to evaluate ZCS in an environment that forces the system to maintain a behavior containing a long chain of actions [7]. They show that ZCS obtains sub-optimal performances. In [3], Bull and Hurst show that ZCS is able to obtain optimal performances in Woods14 with some specific values of parameters.

XCS has been evaluated in Woods14 by Lanzi [16] in order to analyze generalization mechanisms of this LCS. Lanzi obtained results showing that XCS performances do not tend to optimal. On the contrary, the system tends to stay blocked in certain states of the environment.

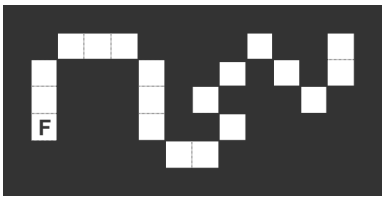

Fig. 6 The Woods14 environment.

ACS has been tested in Woods14 by Butz [5]. The performances obtained in those experiments are optimal. Nevertheless, the parameters used by Butz differs from those used in our experiments[1].

Our prime objective is to observe how imitation, after guidance by a mentor, can improve performance obtained by systems without guidance. For this reason, we use an optimal mentor here. Each experiment begins by a guidance phase of 1000 problems, followed by an exploration phase of 1000 problems, and finally an exploitation phase of 8000 problems. The LCSs' performances without guidance are obtained with an experiment composed of an exploration phase of 1000 problems, followed by an exploitation phase of 9000 problems. Table 3 shows the performances obtained. They are means of all the performances obtained during the last 5000 problems. Figures 7a, 7b and 8 present the steps to food dynamics for all the LCSs and all the approaches. In all the experiments involving XCS the error threshold ϵ_0 is set to 0.05.

	ZCS			XCS			ACS		
Approach	stf	σ_{stf}	im	stf	σ_{stf}	im	stf	σ_{stf}	im
No guidance	9.5	0.23	-	348	13.50	-	13.9	0.56	-
GO	9.5	0.26	100	233	12.94	11	13.9	0.48	76
ABE	9.5	0.24	100	10.5	0.37	91	13.8	0.50	77
IA	9.5	0.24	100	10.1	0.26	95	13.3	0.58	79

Table 3 Mean performances in Woods14. "stf": steps to food performance. "σ_{stf}": standard deviation of the steps to food performance. "% im": percent of action imitating the mentor's behavior.

We may observe that ZCS develops an optimal behavior without guidance. Those results appear surprising at first sight because the parameters we used in our experiments are the ones used by Cliff and Ross who did not obtain such performances. However, the organization of problems in our experiment is different from the one of Cliff and Ross. Indeed, Cliff and Ross' experiments contain only exploration problems. They do not switch to

[1] Butz uses the following parameters: $\gamma = 0.95$, $b_r = 0.05$, $b_q = 0.05$, $\theta_r = 0.9$ and $\theta_i = 0.1$.

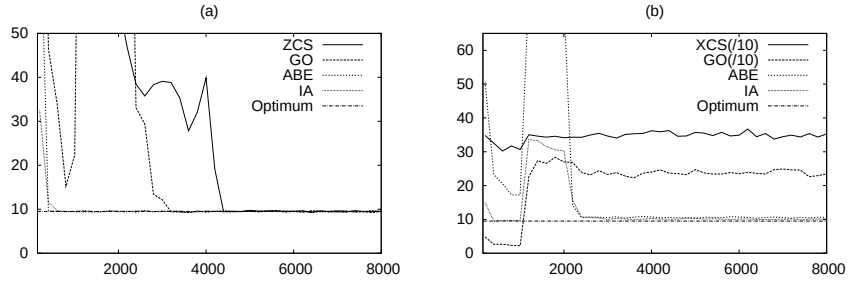

Fig. 7 Steps to food performances in Woods14. (a) Performances with ZCS. (b) Performances with XCS.

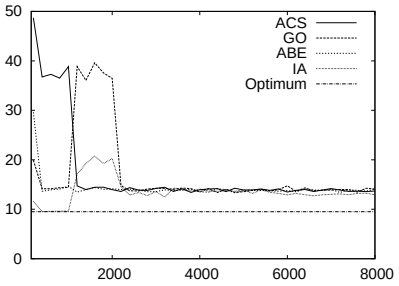

Fig. 8 Steps to food performances in Woods14 with ACS.

exploitation problems and compute their performances during exploration problems. ZCS develops at least an optimal policy in our configuration. Nevertheless, the convergence toward such performances is obtained only after 4000 exploitation problems, i.e. 2000 problems after the end of the exploration phase. The GO approach allows to reach the optimum after 3000 problems while the ABE and IA approaches enable to reach it during the guidance phase, in less than 500 exploitation problems.

Concerning XCS, the classifier system develops some policies which have a mean number of steps of 348 without guidance and 233 with the GO approach (and guidance). The ABE and IA approaches allow to highly improve these performances: after stabilization, the steps to food performance of the ABE approach is 10.5, while the one of the IA approach is 10.1.

Without guidance, ACS tends to a sub-optimal mean number of steps and we can observe that the GO, ABE and IA approaches stay close to this result. The performances obtained without guidance are different from Butz's results in Woods14, but this may be explained by a difference of parameterizing between our system and Butz' experiments. In order to verify this hypothesis, some complementary experiments using the parameters values of Butz have been performed. Optimal performances have then been obtained as in Butz's results.

4.4 Experiments in Woods7

Woods7 has been used by Wilson to evaluate ZCS [32]. The mean number of steps of an optimal policy, according to Wilson, is 2.2. Nevertheless, this environment is non-Markov for systems using a perception limited to the eight neighboring cells. Indeed, such a perception doesn't allow to distinguish every state of the environment i.e. some distinct states are perceived like identical by our agent. The presence of such perceptual aliasings makes impossible for our system to develop an optimal strategy in Woods7. Thus, the mean number of steps obtained by ZCS, in Wilson's experiments, is close to 5 steps to reach the food.

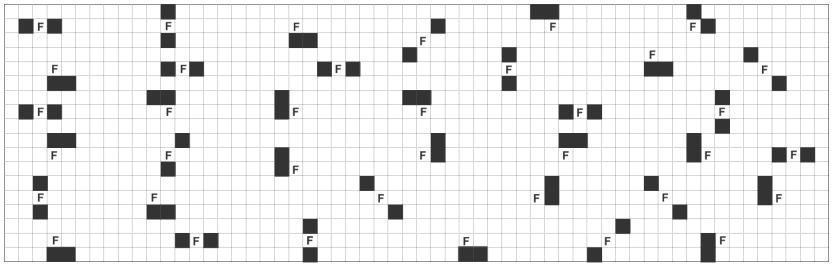

Fig. 9 The Woods7 environment.

Woods7's main characteristic is that it does not represent an impossible problem for our system despite the presence of perceptual aliasing. Indeed, some deterministic policies enable to reach the food starting from any state of the environment and without using any memory mechanism of the past. Thus, the objective of the experiments in Woods7 is to test in what the imitation can help our agent to find and apply those deterministic strategies. In our experiments, the mentor $M1$ applies the following deterministic strategy:

- if the mentor perceives a food, it directly goes to it,
- otherwise, if he perceives an obstacle, he moves to the first free cell preceding the obstacle cell in the order of the perception (turning clockwise and beginning by the north cell),
- otherwise, he moves to the east cell.

The mean steps to food of this policy is 3.70.

Each experiment begins with a guidance phase of 1000 problems, followed by an exploration phase of 1000 problems, and finally an exploitation phase of 8000 problems. The LCSs' performances without guidance are obtained with an experiment composed of an exploration phase of 1000 problems, followed by an exploitation phase of 9000 problems. Table 4 shows the performances obtained. They are means of all the performances obtained

	ZCS			XCS			ACS		
Approach	stf	σ_{stf}	im	stf	σ_{stf}	im	stf	σ_{stf}	im
No guidance	4.97	0.47	-	3.81	0.18	-	4.26	0.17	-
GO	4.97	0.59	46	3.99	0.20	38	3.86	0.19	42
ABE	4.07	0.16	92	5.20	1.25	72	3.87	0.23	88
IA	3.81	0.11	99	3.60	0.14	68	3.69	0.15	85

Table 4 Mean performances in Woods7 with the $M1$ mentor. "stf": steps to food performance. "σ_{stf}": standard deviation of the steps to food performance. "% im": purcent of action imitating the mentor's behavior.

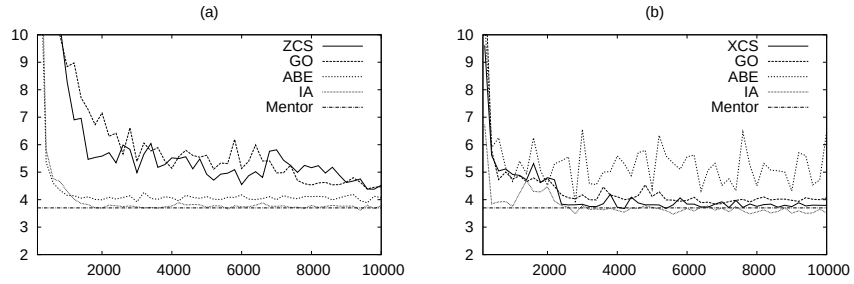

Fig. 10 Steps to food performances in Woods7 when the mentor is $M1$. (a) Performances with ZCS. (b) Performances with XCS.

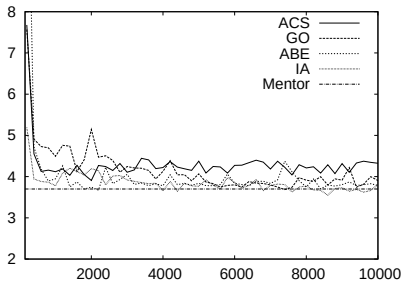

Fig. 11 Steps to food performances with ACS in Woods7 when the mentor is $M1$.

en

during the last 5000 problems. Figures 10a, 10b and 11 present the steps to food dynamics for all the LCSs and all the approaches.

The mean number of steps used by ZCS to reach the food is 4.97. The GO approach does not improve this performance. With the ABE and IA approaches, we may observe that the performances get closer to the mentor's ones. The mean performance of the ABE approach is 4.07, while the one of

the IA approach is 3.81. In parallel, the mentor's behavior is imitated with a mean percent of 92% for the ABE approach, and 99% for the IA approach.

Without guidance, XCS tends to a performance of 3.81. The GO and ABE approaches does not permit to improve this performance. On the contrary, the GO approach develops a behavior with a mean performance close to 4 steps to food, the mentor being imitated for only 38% of the actions. With the ABE approach the mean number of steps is 5.26. Nevertheless, this mean is not really representative of the system performances as they are not steady. The high value of the standard deviation confirms it. This unsteadiness comes from the fact that in some experiments, the system stays locked in a cycling behavior between certain states. Those deadlock situations, however, have appeared on rare occasions. That explains why the performances remain below 10 steps. Finally, the IA approach is the only one whose performance is really better than XCS without guidance. Moreover, it reaches a better number of steps than the mentor, with 3.60 steps against 3.70 for the mentor. The behaviors developed with that approach reproduce 68% of the mentor's behavior.

Applied to ACS, the GO, ABE and IA approaches obtain better performances than without guidance. The IA approach obtains a mean number of steps lightly better than the mentor's one. Although this performance is almost the same as the mentor's, the imitation percentage is close to 85%.

Behaviors of XCS and ACS with the IA approach. The IA approach enabled XCS and ACS to obtain better performances than the mentor. The observation of the policies developed by the two LCSs enabled us to point out several differences between the LCSs' behaviors and the mentor's one.

In most of the states where an obstacle or a food is perceived, XCS and ACS tend to reproduce the mentor's behavior. Thus, the system moves directly to the food if a food is perceived, or circumvents the first obstacle perceived "by the left" otherwise. In some few states, however, we observed a change. Most often those changes bring an improvement to the performances. For instance, the system will circumvent the first obstacle "by the right" or circumvent another obstacle than the first one. The figure 12 shows some examples of the observed changes.

Fig. 12 Examples of policy improvements in Woods7. The full line represents the mentor's actions and the dotted line the system's actions.

With both XCS and ACS, most of the actions that differ from the mentor's behavior are performed in the states where no obstacle and no food is perceived. Despite the fact that the mean performances are rather steady, the behaviors in those states periodically change over the sequence of problems. To understand that phenomenon, it is important to note that all these states are perceived as a unique "empty situation". The Markov policy developed by the LCSs can only provide one unique action for all these states. That action has thus to enable the system to reach as soon as possible a state where an obstacle or a food is perceived. The empty situation is often encountered in Woods7 and thus the classifiers matching the corresponding situation-action pair are often updated. Because any action performed in the empty situation may have a high variety of consequences, the prediction value of those classifiers may often vary. A consequence is that, sometimes, the value of the action of the current policy in the empty situation becomes lower than the value of another action, and so the policy of the system changes. From then on, the fact that XCS obtains better performances than ACS depends on the actions each system tends to select instead of the east action of the mentor. Thus, we observed that XCS most often executes south-east or north-west action. ACS, on the other hand, frequently selects one of those two actions but not as much often as XCS. The specificity of those two actions is to be globally more efficient than east action to enable to reach a state where an obstacle or a food is perceived[2]. That explains, therefore, why XCS obtains better performances than the mentor while ACS stays rather close to him.

Experiences with a new mentor $M2$. The following experiments have been made to test how a mentor performing the best action in the empty situation will impact the performances of our systems. To have a complete view of the effect, the experiments have been carried out with the three LCSs and all our approaches.

In these experiences, the mentor $M2$ follows the policy of $M1$ except in the empty situation where he moves to the south-east cell. The mean steps to food of this policy is 3.35. The organization of phases and the parameters used are the same as in the experiences with $M1$.

The obtained results are presented in the table 5. They show an improvement of all the performances in comparison to the one obtained with the mentor $M1$.

With ZCS, the quality order of the approaches stays the same: the GO approach has a mean performance of 4.83, while the ABE approach obtains 3.70 and the IA one 3.53. As with $M1$, the IA approach enable ZCS to reproduce 98% of mentor's actions.

[2] A system applying the policy of $M1$ around the obstacles and food but performing another action in the empty situation would have the following performance: 3.70 with E or W actions, 3.35 with SE or NW, 3.80 with SW or NE. N and S actions may conduct to cycling behaviors.

	ZCS			XCS			ACS		
Approach	stf	σ_{stf}	im	stf	σ_{stf}	im	stf	σ_{stf}	im
No guidance	4.97	0.47	-	3.81	0.18	-	4.26	0.17	-
GO	4.83	0.35	41	4.22	0.19	41	3.88	0.22	38
ABE	3.70	0.14	92	3.71	0.56	80	3.26	0.09	96
IA	3.53	0.10	98	3.45	0.12	84	3.51	0.16	77

Table 5 Mean performances in Woods7 with the mentor $M2$. "stf": steps to food performance. "σ_{stf}": standard deviation of the steps to food performance. "% im": purcent of action imitating the mentor's behavior.

With XCS, we can observe several changes in comparison with the performances obtained with $M1$. First of all, the ABE approach enables now an improvement of the performances of XCS without guidance. The standard deviation is half of the deviation observed with $M1$ but still stays important in comparison with the deviation obtained with the other approaches. Secondly, the IA approach still has the best performances for XCS, but no more has better performances than the mentor. In fact, XCS still periodically changes the action it performs in the empty situation. For that reason, the system cannot reach performances as good as the $M2$ mentor. As a matter of fact, the mentor performing the best action in the empty situation, the periodical execution of another action can only have some negative effects on the performances.

With ACS, we also observe some changes. As a matter of fact, the ABE approach permits now to obtain better performances than the mentor while the IA approach does not enable it. The reason of the last observation is the same as for XCS. With the IA approach, ACS does not always perform the south-east action in the empty situation. As for XCS, this can only have some negative effects on the performances. Concerning the ABE approach, we can observe that the system reproduces 96% of the mentor's actions. The system reproduces a high majority of the mentor's actions but has find some situations where other actions are more efficient. In fact, the major advantage brought by the ABE approach is to have enabled ACS to always perform the south-east action in the empty situation. It becames possible because the mentor's model computes its own values and imposes them to the main LCS if they are higher than the values computed by that main LCS. In that specific configuration, it enables ACS to maintain the south-east action value higher than the other action values in the empty situation.

5 Discussion

Among our approaches, the GO approach allows us to observe the effect of guidance as a method to explore the environment. Thus, it allows to observe how an exploration of the environment based on a deterministic behavior, mostly optimal, can assist the system's learning. To obtain an

efficient learning in the RL systems, we know that it is necessary to let the system explore the environment long enough before switching to a pure exploitation of its knowledge. The exploration method is of great importance because it determines the way the transitions of the environment are achieved. Therefore, one might expect the guidance by an optimal mentor to harm the performances of the system. This is only partially true. In fact, we observe in our results that the GO approach allows to improve the performances in many of our experiments where optimal performance were not reached without guidance. This result is obtained even when the mentor is not optimal. Despite those improvements, optimal performances could not be obtained in the GO approach if they had not already been obtained without guidance. In the case of XCS in Woods14, the Go approach does not solve the system's difficulties and the performances remain very poor.

The results obtained with the ABE approach vary according to the environment and the type of LCS considered. In Markov environments, the ABE approach is able to improve the performances of the systems in most cases. It enables ZCS to obtain optimal performances in Woods1 if the mentor is optimal. In the non-Markov environment, the ABE approach has allowed to improve the learning capacities of ZCS and ACS. Performances are then close to the mentor's which guided the system except in one case where it enabled ACS to obtain better performances than the mentor. With XCS, however, the system has encountered some deadlock situations. Those deadlocks come from the presence of perceptual aliasing in the environment: a same action in a same situation may lead to very different rewards because some situations may represent several different states. Thus, it is possible to have some overvalued reward predictions in the mentor model. But the mentor model is no longer updated once the guidance period is over. Therefore, a model of mentor containing overvalued predictions after the guidance may act very negatively upon the system's behavior until imposing to it a cycling behavior, as we saw it with XCS in Woods7.

The IA approach is the one that has achieved the best performances in most of the environment/system/mentor configurations. The AI approach enables to improve the performances of our systems in every tested configurations where an improvement was possible. The AI approach presents several advantages in comparison to the ABE approach. Especially, it applies a latent learning of the mentor's behavior. This allows the system to model the mentor without taking account of the environmental reward. This allows that approach to be extended to different mentor/observer interaction models because it is not necessary to test the mentor's actions in the environment to build the behavioral model. Another advantage of this approach is that an erroneous model of mentor's behavior will not have any blocking influence on the system's global behavior.

6 Conclusion

In this paper, we presented three approaches of imitation for the classifier systems architectures. Our objective was to enable an observer agent to improve its performances by taking into account the behavior of a mentor agent. Those approaches have been evaluated and compared in different environments as they were applied to the three major learning classifier systems today: ZCS, XCS and ACS. Although the environments used have a small size, they allow us to highlight some essential features of their effect on learning.

The results showed, in several occasions, an improvement of the classifiers systems performance by taking into account an observed behavior. Moreover, those results showed why it was relevant to use an explicit mentor's model in addition to the classifier system. Regarding this point, two models have been considered. In the first case, we used a rewards model, specific to the mentor's behavior, whose predictions are integrated into the the action selection process through a maximization. That first model enabled to efficiently take account of the observed behavior in Markov environments. In non-Markov environments, however, that approach may cause some deadlock phenomena. The second model relies on a behavioral model of the mentor. The outputs of that model are used in the action choice through the use of an additional action called imitation action. That approach achieved the best performances in the most of the tested configurations. That model appears more efficient than the former one on several points. It especially enables a latent learning of the mentor's behavior and is not subjected to deadlock phenomenons in non-Markov environments.

The study presented in this paper compares several approaches. It highlights the relevance of an observed behavior being taken into account and shows that the use of an imitation action is an efficient method. However, it does not highlight how those mechanisms enable to improve the system's abilities. The next logical step of our study would be to pursue the study of the imitation action approach in order to analyze more precisely the improvement brought to the system. Some other perspectives may consist in extending the architectures of our approaches. It might be possible to develop some mechanisms enabling to avoid the deadlock phenomenons of the approach based on a mentor reward model. For instance, we might use a *reliability* parameter in order to measure the accuracy of the predictions of the mentor model. Another perspective might be to allow the use of generalization in the mentor models. Finally, the logical continuation of this work will be to study imitation for learning classifier systems in environments of increasing complexity in order to tend toward large dynamic multi-agent environments.

References

1. Christopher G. Atkeson and Stefan Schaal. Robot learning from demonstration. In *Proceedings of the 14th International Conference on Machine Learning*, pages 12–20. Morgan Kaufmann, 1997.
2. P. Bakker and Y. Kuniyoshi. Robot see, robot do: An overview of robot imitation. In *AISB'96 Workshop on Learning in Robots and Animals*, pages 3–11, Brighton, UK, 1996.
3. Larry Bull and Jacob Hurst. ZCS Redux. *Evolutionary Computation*, 10(2):185–205, 2002.
4. Martin Butz, David E. Goldberg, and Wolfgang Stolzmann. New challenges for an ACS: Hard problems and possible solutions. Technical Report 99019, University of Illinois at Urbana-Champaign, Urbana, IL, October 1999.
5. Martin Butz, David E. Goldberg, and Wolfgang Stolzmann. The Anticipatory Classifier System and Genetic Generalization. Technical Report 2000032, Illinois Genetic Algorithms Laboratory, 2000.
6. Martin V. Butz and Stewart W. Wilson. An algorithmic description of XCS. *Soft Computing*, 6:144–153, 2002.
7. Dave Cliff and Susi Ross. Adding temporary memory to zcs. *Adaptive Behavior*, 3(2):101–150, 1994.
8. B. F. Damer. *Avatars, Exploring and Building Virtual Worlds on the Internet*. Addison-Wesley Longman/Peachpit Press, 1998.
9. M. Dorigo and M. Colombetti Robot Shaping: Developing Autonomous Agents through Learning. *Artificial Intelligence*, 71:321–370, 1994.
10. David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Mass., 1989.
11. Jean-Claude Heudin. Virtual Worlds. In J.C. Heudin, editor, *Virtual Worlds: Synthetic Universes, Digital Life and Complexity*, pages 1–28. Perseus Books, 1998.
12. Joachim Hoffmann. *Vorhersage und Erkenntnis [Anticipation and Cognition]*. Hogrefe, 1993.
13. John H. Holland. Escaping Brittleness: The possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In Mitchell, Michalski, and Carbonell, editors, *Machine learning, an artificial intelligence approach. Volume II*, chapter 20, pages 593–623. Morgan Kaufmann, 1986.
14. Tim Kovacs. Evolving Optimal Populations with XCS Classifier Systems. Master's thesis, School of Computer Science, University of Birmingham, Birmingham, U.K., 1996. Also technical report CSR-96-17 and CSRP-96-17.
15. Tim Kovacs and Manfred Kerber. What makes a problem hard for XCS? In Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors, *Advances in Learning Classifier Systems*, volume 1996 of *LNAI*, pages 80–99. Springer–Verlag, Berlin, 2001.
16. Pier Luca Lanzi. An Analysis of Generalization in the XCS Classifier System. *Evolutionary Computation*, 7(2):125–149, 1999.
17. Pier Luca Lanzi. Learning classifier systems from a reinforcement learning perspective. *Soft Computing*, 6(3):162-170, 2002.
18. L.-J. Lin. Self-improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching. *Machine Learning Journal*, 8:293-321, 1992.
19. Maja J. Mataric. Using communication to reduce locality in distributed multiagent learning. *J. of Experimental and Theoret. Artif. Intell.*, 10(3):357–369, 1998.

20. R. W. Mitchell. A comparative-developmental approach to understanding imitation. *Perspectives in Ethology*, 7:183–215, 1989.
21. Andrew Y. Ng and Stuart Russell. Algorithms for Inverse Reinforcement Learning. In *Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, San Francisco, CA, 2000.
22. Bob Price. *Accelerating Reinforcement Learning with Imitation*. PhD thesis, University of British Columbia, 2003.
23. Bob Price and Craig Boutilier. Implicit imitation in multiagent reinforcement learning. In *Proc. 16th International Conf. on Machine Learning*, pages 325–334. Morgan Kaufmann, San Francisco, CA, 1999.
24. C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In *Proceedings of the Ninth International Conference on Machine Learning*, Aberdeen, July 1992. Morgan Kaufmann.
25. Wolfgang Stolzmann. Anticipatory classifier systems. In *Proceedings of the Third Annual Genetic Programming Conference*, pages 658–664. Morgan Kaufmann, 1998.
26. Dorian Šuc and Ivan Bratko. Skill reconstruction as induction of LQ controllers with subgoals. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 914–919, San Francisco, August 23–29 1997. Morgan Kaufmann Publishers.
27. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, 1998.
28. Tanja Urbančič and Ivan Bratko. Reconstructing Human Skill with Machine Learning. In A. G. Cohn, editor, *Proceedings of the Eleventh European Conference on Artificial Intelligence*, pages 498–502, Chichester, August 8–12 1994. John Wiley and Sons.
29. Paul E. Utgoff and Jeffrey A. Clouse. Two Kinds of Training Information For Evaluation Function Learning. In Dean, McKeown, editor, *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 596–600. MIT Press, July 1991.
30. Christopher JCH Watkins and Peter Dayan. Q-Learning. *Machine Learning*, 8(3):272–292, 1992.
31. Steven D. Whitehead. A Complexity Analysis of Cooperative Mechanisms in Reinforcement Learning. In Dean, McKeown, editor, *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 607–613. MIT Press, 1991.
32. Stewart W. Wilson. Zcs: A zeroth level classifier system. *Evolutionary Computation*, 2(1):1–18, 1994.
33. Stewart W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.