Adaptive Agents with Reinforcement Learning and Internal Memory

Pier Luca Lanzi
Artificial Intelligence and Robotics Laboratory
Dipartimento di Elettronica e Informazione
Politecnico di Milano
lanzi@elet.polimi.it

Abstract

Perceptual aliasing is a serious problem for adaptive agents. Internal memory is a promising approach to extend reinforcement learning algorithms to problems involving perceptual aliasing. In this paper we investigate the effectiveness of internal memory for tackling perceptual aliasing problems with adaptive agents and reinforcement learning. Specifically, we try to give a unified view of some interesting results that have been presented in different frameworks, i.e.: tabular reinforcement learning and learning classifier systems.

1. Introduction

The learning capabilities of *adaptive agents* rely heavily on their perception of the environment. There are cases in which the agent can perceive all the relevant information necessary to decide what is best to do in every situation. Often the agent's perceptions are aliased because the agent's sensors convey only *partial* information about the surrounding environment. In these cases there might be *distinct* situations which appear *identical* to the agent but require different actions. When this happens the agent cannot decide what is best to do only looking at its current sensory inputs since these may be aliased. Shortly, we say that the agent suffers from a *perceptual aliasing problem*.

Perceptual aliasing is a major problem in reinforcement learning (Sutton and Barto 1998), that area of machine learning devoted to the study of techniques to train autonomous agents through trial and error interactions. In fact most reinforcement learning techniques assume that the environment is completely observable by the agent, i.e., that the agent's perceptions are not aliased. This condition is usually termed the *Markov* property. The distinction between environments which satisfy the Markov property (i.e., *Markov environments*) and those which do not (i.e., *non-Markov environments*) is essential in reinforcement learning.[1] If the environment is Markov, the agent can rely completely on its current sensations, and those reinforcement learning algorithms which develop reactive behaviors (e.g., Watkins' Q-learning (1989)) perform optimally. Conversely, if the environment is non-Markov, the agent cannot rely completely on its current sensations because these might be aliased; reactive behaviors are not optimal anymore and most of the reinforcement learning algorithms become infeasible. To cope with perceptual aliasing, reinforcement learning algorithms are usually extended by adding a memory mechanism. Since the agent cannot rely only its current sensory inputs, it exploits memory to keep trace of previous information to disambiguate current situations.

One of the most promising approaches to add memory to adaptive agents is by means of an internal memory register (Wilson 1994) in which the agent can read and write information. From the agent's point of view internal memory is a sort of knot in an handkerchief that the agent can tie or untie to remember past events. The results reported in the literature suggest that internal memory[2] can be a *general* and *effective* way to tackle perceptual aliasing. However, there are small discrepancies between the results reported by different authors. Lanzi (1998b), and later Lanzi and Wilson (2000), discussed a number of *general* problems that show up when using internal memory to solve perceptual aliasing. Peshkin et al. (1999) suggested that SARSA(λ) easily reached optimal performance. Unfortunately these results refer to different testbeds so that a direct comparison between the two approaches is impossible.

In this paper we investigate the effectiveness of in-

[1] The term non-Markov embraces a larger class of problems than that captured by the term "perceptual aliasing". However, since in this paper we only consider problems that are non-Markov because of perceptual aliasing we use these two terms without distinctions.

[2] In learning classifier systems both Holland (1986) and Wilson (1994) consider memory as *internal* to the agent; in contrast Peshkin et al. (1999) consider memory as *external*. In this paper we use use the former approach which is coherent with Sutton and Barto (1998) who consider as *internal* anything that the agent can explicitly modify.

ternal memory for solving perceptual aliasing within the reinforcement learning framework. Our aim is to present an initial step toward the development of a unified view of different results reported in the literature. For this purpose, we consider the most known reinforcement learning algorithm, Q-learning, and add internal memory following the approach used in (Lanzi and Wilson 2000; Peshkin et al. 1999). We apply Q-learning with internal memory to a problem involving simple perceptual aliasing. We show that while trying to solve aliasing *on the agent's perceptions* we may introduce an aliasing *on the payoffs* which, in principle, prevents the development of satisfactory behaviors. We study the conditions under which this aliasing *on the payoffs* can be limited in order to allow satisfactory performance. We show that to limit this phenomenon the agent should not develop a *complete* knowledge of the underlying reinforcement learning problem but a *partial description*. Then we consider three extensions introduced in the literature for improving the performance of internal memory within reinforcement learning: (i) the separation between the exploration of internal memory and the exploration of the environment (Lanzi 1998b); (ii) the use of redundancy (Lanzi and Wilson 2000); (iii) the eligibility traces (Peshkin et al. 1999). Finally we discuss the impact of these three extensions on Q-learning with internal memory with respect to the problem previous discussed.

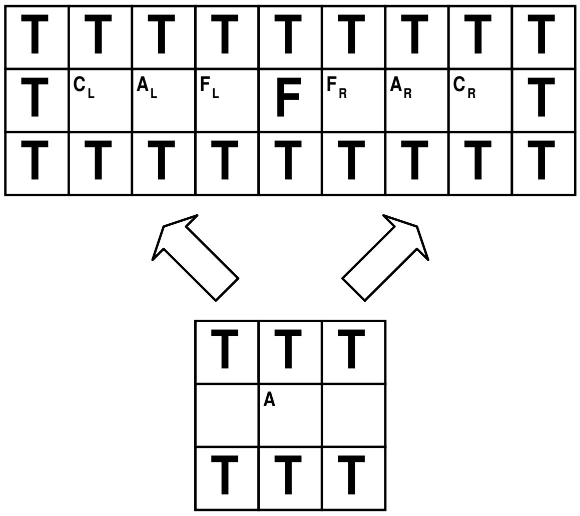

Figure 1: The Woods100 environment. The labels C_L, A_L, F_L, F_R, A_R, and $C_R$identify the empty positions. Positions $A_R$and $A_L$are aliased since they are perceived as identical (like A) by the animat.

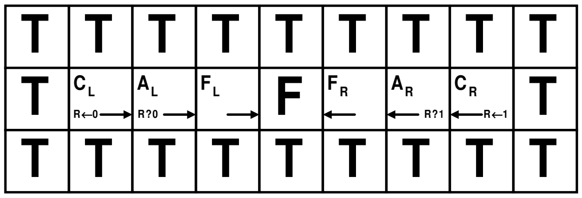

Figure 2: An optimal policy for Woods100 with one bit on internal register. With "R←. . ." we indicate that the register is set *before* the animat moves; with "R?. . ." we indicate that the animat moves only if the register contents are set as specified.

2. Related Work

Internal memory was first introduced within the *biologically-inspired* framework of learning classifier systems (Holland 1986). These are rule-based systems in which learning is viewed as the process of ongoing *adaptation* to a partially unknown environment through *genetic algorithms* and temporal difference learning. Learning classifier systems have an *internal message list* where the agent can store and retrieve information about previous time steps. Wilson (1994) simplified Holland's approach by replacing the internal message list with an *internal memory register* which basically corresponds to an internal message list with only one message. Wilson's internal memory register was successfully applied to many problems involving perceptual aliasing, e.g.: Cliff and Ross (1994), Lanzi (1998a, 1998b), Lanzi and Wilson (2000). Recently Peshkin et al. (1999) applied the internal register idea to the framework of tabular reinforcement learning adding an internal register to SARSA(λ) and to a novel algorithm, named VAPS, reporting interesting results.

Other authors have introduced different approaches to tackle perceptual aliasing with reinforcement learning. McCallum (1995) extensively studied algorithms to exploit a window of previous agent's sensations to identify and solve perceptual aliasing situations. Stolzmann (2000) developed his *anticipatory classifier system* where previous sensations are explicitly remembered and used to disambiguate states that appear to be aliased.

3. Perceptual Aliasing and Memory

To study how internal memory can be exploited to tackle perceptually aliased situations we adopt the biological metaphor of an artificial animal, an *animat* (Wilson 1985), that must learn to find food inside a maze. Environments are grid worlds where each cell can contain trees ("T"), food ("F"), or can be empty. The animat randomly placed in an empty position must reach a position with food. The animat has two Boolean sensors for each of the eight adjacent cells: one tells the animat whether the cell is empty, the other one whether the cell contains food; the animat can move into any adjacent empty cell.

The Woods100 environment in Figure 1 is a small corridor with food in the middle position; it has two positions, identified with the labels A_L and A_R, where the animat's sensations are aliased. In fact these positions appear identical (like position A in the lower part of Figure 1) through the animat's sensors but they require different optimal actions: in position A_L the optimal action is go-east; in position A_R the optimal action is go-west. From now on, with A_L and A_R we will refer to the actual positions in the environment, with A to the animat *sensation* in both A_L and A_R.

Perceptual aliasing in Woods100 would be easily solved if the animat could remember the position from which it entered an aliased position: if the animat was in the left corner (position C_L) before entering the aliased position (i.e., before sensing A), the optimal action is go-east; if the animat was in the right corner (position C_R) before entering the aliased position, the optimal action is go-west.

To remember its previous position the animat can use an internal memory register (R) with one bit. When the animat is initially placed in the environment the register R is set to 0. If the animat is in C_L it will set the internal memory register R to zero, if the animat is in C_R it will set the internal memory register to one; when in an aliased position (i.e., when its sensations are A) the animat will go left if R is one, right if R is zero. This strategy to solve the perceptual aliasing in Woods100 is depicted in Figure 2. Note that when the animat starts in an aliased position (e.g., A_R) it first must find out in which part of the environment it is. For example, if the animat is initially put in position A_R it will first advocate action go-east without setting the register; as a result the animat will find itself in position C_R. At this point the animat realizes that it is in the right side of the corridor; accordingly it goes west setting the register R to 1 entering, for the second time, the aliased position (A_R). At this point, the animat "*remembers*" that it is in the right side of the corridor since the register R is set to one; thus it goes west finding food near position F_R.

4. Reinforcement with Internal Memory

Reinforcement learning addresses the problem of an *agent* that must learn to perform a task through *trial and error* interactions with an unknown *environment* (Sutton and Barto 1998). The agent and the environment interact continuously. The agent senses the environment through its *sensors* and, based on its current sensory inputs, it selects an *action* to perform in the environment. Depending on the effect of its action, the agent eventually obtains a *reward*. The agent's goal is to *maximize* the amount of reward (i.e., the *payoff*) it receives *in the long run*.

There exist different types of algorithms that can tackle reinforcement learning tasks. Most of them are inspired, in a way or another, by methods of Dynamic Programming (e.g., Watkins' Q-learning (1989)). Others, like Holland's learning classifier systems are more inspired by techniques of adaptive behavior.

Watkin's Q-learning (1989) is probably one of the best known reinforcement learning algorithm. The Q-learning algorithm computes by successive approximations a table of all values $Q(s,a)$, called the Q-table. $Q(s,a)$ is defined as the payoff that the agent should expect when, sensing s it performs action a, and then it carries on always performing the best possible actions. The algorithm works as follows. Initially, all the values $Q(s,a)$ are randomly initialized. At each time step t, when action a_t is performed in state s_t the Q-table Q is updated according to the formula:

$$Q(s_{t-1},a_{t-1}) \leftarrow \\ \alpha Q(s_{t-1},a_{t-1}) + (1-\alpha)(r + \gamma \max_a Q(s_t,a))$$

where α is the *learning rate* ($0 \leq \alpha \leq 1$); γ is the *discount factor*; r is the reward received for performing a_{t-1} in s_{t-1}; and s_t represents the agent's sensations after performing a_{t-1} in s_{t-1}.

Action Selection. Every time the agent has to select an action it can decide whether it should *exploit* what it already knows or it should *explore* new possibilities. In the former case the agent usually selects the action with the highest expected reward. In the latter the agent selects an action according to a certain policy. There are many policies that an agent can follow to select an action to be performed in the environment. In this paper the animat uses an ϵ-greedy policy to explore new actions: with probability ϵ the animat will explore the environment by selecting a random action; with probability $1-\epsilon$ the animat will select the action with the highest expected reward.

Eligibility Traces. Tabular Q-Learning can be extended with Sutton's *eligibility traces* (Sutton 1988). The resulting algorithm, Q(λ), at each time step t performs the following updates (Watkins 1989):

1. $\delta \leftarrow (r + \gamma \max_a Q(s_t,a)) - Q(s_{t-1},a_{t-1})$
2. $e(s_{t-1},a_{t-1}) \leftarrow e(s_{t-1},a_{t-1}) + 1$
3. `for all` s,a:
   $Q(s_{t-1},a_{t-1}) \leftarrow Q(s_{t-1},a_{t-1}) + \alpha\delta e(s,a)$
   `if` a_t `is the best action`
   `then` $e(s,a) \leftarrow \gamma\lambda e(s,a)$;
   `otherwise` $e(s,a) \leftarrow 0$

where $e(s,a)$ is the eligibility trace corresponding to sensation s and action a; the term λ is a decay factor. Q(λ) is generally believed to outperform plain Q-learning since it uses single experiences to update the estimates of many other sensation-action pairs (Peng and Williams 1996; Wiering and Schmidhuber 1998a). In addition, eligibility traces are often used when dealing with non-Markov environments, e.g., Loch and Singh (1999) and Peshkin et al. (1999).

Q-Learning with Internal Memory. To add internal memory to Q-learning we follow the approach proposed by Wilson (1994) for the ZCS classifier system (later implemented by Cliff and Ross (1994) and Lanzi (1998a)) which is is basically the same used by Peshkin et al. (1999) with SARSA(λ). First we add an internal memory register R of b bits to Q-learning. We extend the animat's sensations s with a binary string s_m that the animat can use to test the content of the register, i.e., s_m is satisfied if R is equal to s_m. We also extend the animat's actions with an "*internal action*" a_m which the animat can use to modify the register, i.e., to perform a_m means to assign a_m to R. At the beginning of a new problem the register is set to a fixed initial state (e.g., R is set to zero). Then Q-learning is applied as usual on the Q-table built on the new set of sensation pairs (s,s_m) and action pairs (a,a_m). The only difference with respect to *basic* Q-learning is that: (i) the animat sensations are determined both by the current (external) sensations s and by the content of the register R; (ii) the animat acts both on the environment (with a) and on the internal register R (with a_m).

Design of Experiments. All the results presented in this paper are averages over one hundred runs. Each runs consists of 50000 problems that the animat must solve. These problems are divided in *learning problems* and *testing problems*: every learning problem is followed by a testing problem. During a learning problem the animat selects actions randomly according to the ϵ-greedy exploration strategy, during testing problems the animat always selects the best action. The animat performance is computed as a moving average over the last 50 testing problems. At the end of the 50000 problems, the solution is tested by running a set of test problems in which the animat is put in every free position of the environment exactly once.

5. Simple Perceptual Aliasing

We apply Q-learning with one bit of internal memory (i.e., $b = 1$) to Woods100 for 50000 problems, with the following parameter settings: $\alpha = .1$, $\gamma = .5$, $\epsilon = .5$. The performance, computed as an average over one hundred experiments, is showed in Figure 3. As can be noticed from the plot, Q-learning with one bit of internal memory does not achieve optimal performance on the average. However, when looking at the single runs we find that:

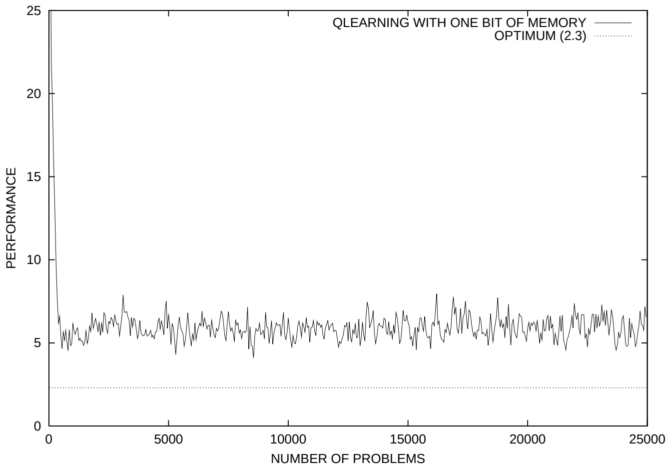

Figure 3: Performance of Q-learning with *one* bit in Woods100. The curve is an average over one hundred runs.

(i) in *62* runs Q-learning with memory converged to an optimal solution for Woods100 which was the same as or equivalent to that shown in Figure 2;

(ii) in other *20* runs Q-learning with memory converged to a solution that was "*correct*", i.e., that always took the animat to the food but required more steps than the optimal solution in Figure 2;

(iii) in the remaining *18* runs Q-learning with memory converged to an incomplete solution which was fully incorrect, e.g., for certain positions the animat bounces against a wall or loops in a subset of positions.

We repeated this experiment for values of ϵ from .1 to 1.0 with step .1. The result of this series of experiments is depicted in Figure 4. The parameter ϵ is on the abscissa; the percentage of *optimal* solutions (dashed line) and *correct* solutions (solid line) is on the ordinate. As the results show in *no case* Q-learning with one bit of memory reaches optimal performance in 100% cases.

6. Analysis of the Experimental Results

The reader might argue that the unsatisfactory performance of Q-learning with memory in Woods100 is due to an incorrect use of the memory bit. However analyzing these results in detail we noted that in most of the cases in which performance was not optimal the association between the memory bit and the aliased positions was correct. In fact in many unoptimal cases, the solution for the positions A_R, A_L, F_L, and F_R was similar to that in Figure 2.

Surprisingly (at least for us), unoptimal performance happened because the best action advocated in the corner position (C_L and C_R) was wrong. For instance, if

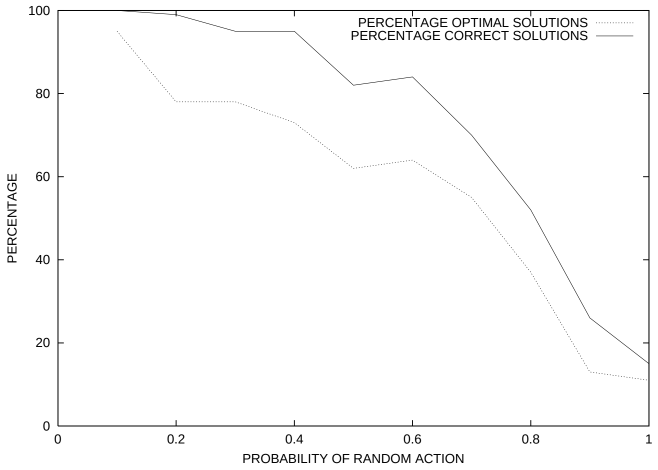

Figure 4: Percentage of *optimal* (dashed) and *correct* (solid) solutions developed by Q-learning with one bit of internal memory for different values of ϵ. The statistic is measured over 100 runs of 50000 trials each.

we consider the solution in Figure 2, in many unoptimal cases we found that in position C_L the action corresponding to the highest expected reward was not go-east setting the register R to 0 (shortly, go-east$_{R\leftarrow 0}$), as it happens in the optimal solution in Figure 2, but go-east setting R to 1 (shortly, go-east$_{R\leftarrow 1}$). As a result, when the animat started in position C_L it cycled between C_L and A_L. This looping situation eventually stopped when, because of the Q-learning update, the *truly* optimal action go-east$_{R\leftarrow 0}$ became the most predictive and the animat broke out of the loop.

7. The Mathematical Framework

To understand the previous results we must first understand the impact that internal memory has on the underlying mathematical framework of Q-learning.

When adding internal memory to Q-learning we hypothesize that, if the environment has some aliased situations (and thus is non-Markov), then the animat can solve perceptual aliasing (making it Markov) by storing and retrieving information in the internal memory. Indeed, the optimal solution for Woods100 in Figure 1 suggests that this is possible and that our hypothesis is correct. On the other hand, our results also suggest that something in the overall framework does not work properly. In fact we found that an unoptimal action (e.g., go-east$_{R\leftarrow 1}$ in C_L) might have an higher expected reward than an optimal one (e.g., go-east$_{R\leftarrow 0}$). Thus the first question we must answer is, given the optimal solution in Figure 2, "What is the expected reward of the actions go-east$_{R\leftarrow 0}$ and go-east$_{R\leftarrow 1}$ in C_L?"

Consider the optimal solution in Figure 2 and suppose that Q-learning with one bit of memory has converged to the optimal Q-table Q^* for Woods100.[3] ¿From the Q-learning we have that the expected payoff for action go-east$_{R\leftarrow 0}$ in C_L, i.e., $Q^*(C_L, \text{go-east}_{R\leftarrow 0})$ is computed as:

$$Q^*(C_L,\text{go-east}_{R\leftarrow 0}) = r + \gamma \max_{a\in A} Q^*(A,a)$$

The immediate reward r is zero since performing go-east$_{R\leftarrow 0}$ in C_L returns no reward; the most rewarding action in the next (aliased) state A[4] is go-east$_{R?0}$ which has an expected payoff equal to $\gamma 1000$ since it is two steps away from the food. Thus $Q^*(C_L, \text{go-east}_{R\leftarrow 0})$ is $\gamma^2 1000$. In a similar way, if we compute $Q^*(C_L,\text{go-east}_{R\leftarrow 1})$ we find that: again r is zero; when the internal memory register R is set to 1, the most rewarding action in A is go-west$_{R?1}$ which is two steps away from the food and thus as a payoff equal to $\gamma 1000$; while $Q^*(C_L,\text{go-east}_{R\leftarrow 1})$ is $\gamma^2 1000$. Therefore the two actions go-east$_{R\leftarrow 0}$ and go-east$_{R\leftarrow 1}$ have the *same* expected reward in C_L that is: $\gamma^2 1000$!

Although in Woods100 the animat can exploit memory to solve the perceptual aliasing coming from its sensors, it may be still unable to decide which action is best because also the action's payoffs are "aliased," i.e., the animat cannot distinguish which action is best because both actions have the *same* expected payoff.

8. Validation

This analysis suggests that the animat should not be able to solve Woods100. On the other hand our experiments show that the animat can solve Woods100 in a significant amount of experiments: when ϵ is 0.5, 62% of solutions developed are optimal (Figure 4). Thus, if Q-learning with internal memory should not solve Woods100, why does the animat learn so many optimal solutions?

To explain these results we traced the payoffs of the actions go-east$_{R\leftarrow 0}$ and go-east$_{R\leftarrow 1}$ in position C_L for different values of ϵ. Figure 5 shows the payoffs of the two actions in position C_L on a single *optimal* run for $\epsilon = .1$; Figure 6 shows the same payoffs for an optimal run with $\epsilon = .5$; finally, Figure 7 shows them for an unoptimal run with $\epsilon = 1.0$.

As can be noticed from Figure 5, when the animat performs only a little exploration ($\epsilon = 0.1$) one of the two actions (solid line in Figure 5) becomes immediately "dominant," i.e., its expected reward is greater than the expected reward of the other action. Thus when the exploration is limited the two actions, go-east$_{R\leftarrow 0}$ and go-east$_{R\leftarrow 1}$, are very *unlikely* to converge to the same payoff, therefore the animat is always able to choose the best action when in C_L.

[3] Since we hypothesized that internal memory solves perceptual aliasing we also hypothesize that the Markov hypothesis holds and that Q-learning with internal memory converges to an optimal Q^*.

[4] We remind the reader that we use A to refer to the aliased positions as seen through the animat's sensations; A_L and A_R when referring to the actual positions in the environment.

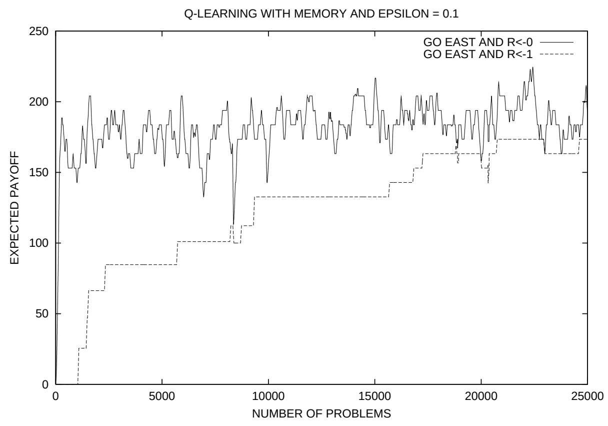

Figure 5: Values of the Q-table corresponding to position $\mathsf{C_L}$for the actions $\mathsf{go\text{-}east}_{\mathsf{R}\leftarrow 0}$ and $\mathsf{go\text{-}east}_{\mathsf{R}\leftarrow 1}$; the parameter ϵ is 0.1.

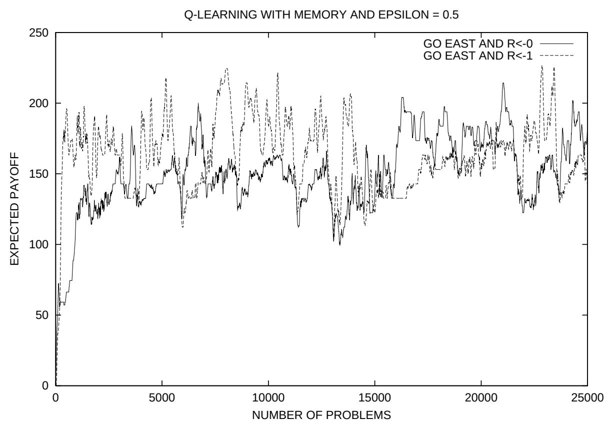

Figure 6: Values of the Q-table corresponding to position $\mathsf{C_L}$for the actions $\mathsf{go\text{-}east}_{\mathsf{R}\leftarrow 0}$ and $\mathsf{go\text{-}east}_{\mathsf{R}\leftarrow 1}$; the parameter ϵ is 0.5.

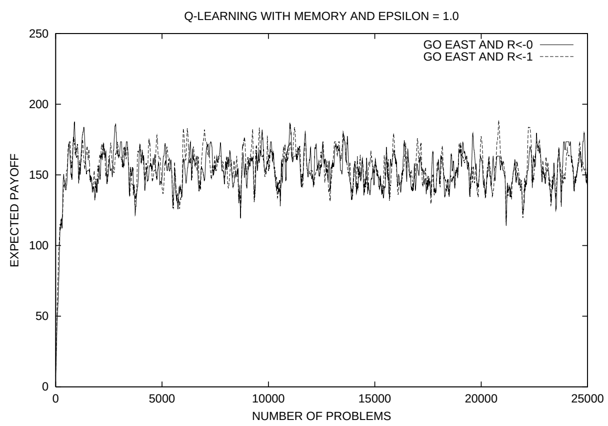

Figure 7: Values of the Q-table corresponding to position $\mathsf{C_L}$ for the actions $\mathsf{go\text{-}east}_{\mathsf{R}\leftarrow 0}$ and $\mathsf{go\text{-}east}_{\mathsf{R}\leftarrow 1}$; the parameter ϵ is 1.0.

When the exploration rate increases ($\epsilon = .5$) the expected payoffs of the two actions in $\mathsf{C_L}$ tend to become *very* similar (see Figure 6). However, one action may still tend to be dominant because it is exploited more (see the dashed line in Figure 6). Note however that the two actions may swap their roles, i.e., the action which has a lower expected payoff may suddenly become the more predictive and *vice-versa*; this event is reported in the plot with the crossing of the two lines (see Figure 6 between run 15000 and run 20000). When this happens two events may occur: (i) If the new situation persists enough the animat will tend to modify the association between the memory and the aliased positions so that the new "dominant" action becomes optimal; (ii) Alternatively, after a few trials the two actions swap again and the former optimal action becomes dominant again.

Finally, if the animat explores too much ($\epsilon = 1$) the prediction of the two actions might become almost equal and the corresponding actions are indistinguishable with respect to their expected payoffs (see Figure 7). Thus the animat cannot establish an optimal and stable strategy to solve perceptual aliasing in Woods100.

9. Some Considerations

We found that while trying to solve aliasing on the animat's perceptions with internal memory, we might introduce an "*aliasing on the payoffs*" which can prevent the animat to develop satisfactory behaviors. Unfortunately, this type of alias is very difficult to characterize. As we will show in the next section, other well know environments show similar situations. One way to tackle this problem consists of performing little exploration. In this case, even if the problem does not admit *in principle* an optimal solution the animat can build one by focusing on most promising actions and paying little or no attention to less promising ones. In other words, to solve perceptual aliasing the animat *must* learn only *part of what it can do* (in our case, only *part* of the Q-table) because the complete knowledge of *everything* it might do (i.e., the complete final Q-table) would make the problem impossible to solve. In reinforcement learning the complete knowledge of the Q-table is usually regarded as an *advantage* of tabular representations. Unfortunately these are not feasible in practice because of their complexity and thus function approximators (e.g., neural networks) are commonly employed to get rid of the size of the Q-table. In general, these methods do not represent the whole table, but only part of it. Our results suggest that these "*incomplete*" representations become *a need* when using internal memory with reinforcement learning.

Finally, we observe that a limited exploration *does not* solve the problem we analyzed. In fact, even if we use a very small exploration rate ϵ we are not guaranteed that the animat will not finally develop Q^* reaching *at the end* an unsatisfactory behavior. In fact, in Figure 5 we

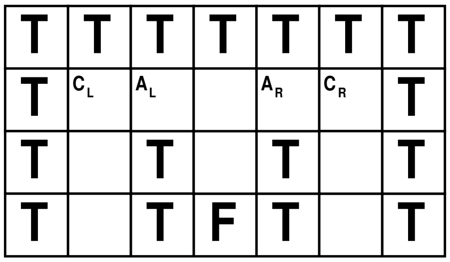

Figure 8: The Woods101 environment. The labels A_L and A_R identify the aliased positions. The labels C_L and C_R identify the corner positions.

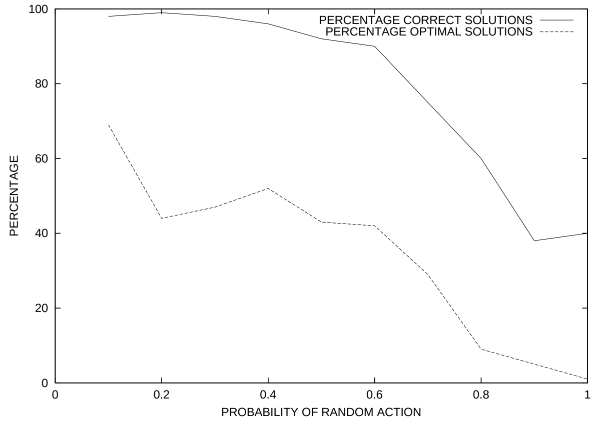

Figure 9: Number of *optimal* (dashed) and *correct* (solid) solutions with one bit of memory and different values of ϵ. The statistics are measured over 100 runs of 50000 trials each.

see that although the animat rarely explores the action with the lower payoff, in the long run we shall expect that the two lines will cross, i.e., the two payoffs will tend to be very, very, similar.

In our opinion, these results suggest that "non-tabular" reinforcement learning techniques, such as learning classifier systems or neural networks, are promising ways to tackle perceptual aliasing problems. These techniques in fact more easily focus only on part of the representation of animat's behavior; this feature when coupled with adequate exploration strategies may avoid the convergence to the complete Q-table and the consequent disruption of the optimal behaviors developed. The positive results presented in literature, e.g., (Lanzi and Wilson 2000) and (Wiering and Schmidhuber 1998b), appear to support this opinion.

10. A Well Known Environment

Since Woods100 is very simple we may argue that the phenomenon we analyzed so far appears rarely. However, if we consider the Woods101 environment which has been extensively used in much reinforcement learning literature (e.g., McCallum (1996), Cliff and Ross (1994)) we find that the phenomenon we analyzed shows up again. Woods101 (Figure 8) is a small maze with two aliased positions (A_L and A_R) which can be solved, in principle, by an animat with one bit of internal memory. The two corner positions (C_L and C_R) have the same problems as the positions with the same name in Woods100 (not showed here). When we apply Q-learning with one bit of internal memory in Woods101 with the same settings of previous experiments, and we measure the number of *optimal* and *correct* solutions developed, we obtain the results showed in Figure 9 which basically confirm what we found for Woods100. Interestingly, as far as we know, the phenomenon we discussed in previous sections has not been discussed in the literature; this supports our claim that this "*aliasing on the payoffs*" may be very difficult to characterize and thus a relevant problem when designing adaptive agents.

11. Three Extensions

Other authors have studied the use of internal memory within the reinforcement learning framework. In these works three main extensions were introduced to improve the performance of reinforcement learning algorithms with internal memory.

Internal/External Exploration. Lanzi (1998b) showed that it is useful to separate the exploration performed "*in the environment*" from that performed "*in the internal memory.*" In particular he suggested that for reaching optimal performance in difficult perceptually aliased situations, it is better to explore *internal* memory less than *external* environment. Lanzi argued that to build a correct association between memory configurations and aliased positions the animat must follow a strategy which must be "*more deterministic*" than that needed to explore the environment.

To analyze the impact of these two types of explorations in our framework we define two exploration probability: ϵ_X for the environment; ϵ_M for the internal memory. We extend the ϵ-greedy exploration employed in the previous experiments as follows. Every time the animat has to select an action, first it selects with probability ϵ_M whether it will act on the internal memory randomly or with the best available action. In the former case, the animat will generate a random (internal) action to be performed in the internal memory register. In the latter case, it will select the internal action that corresponds to the most rewarding external-internal action pair. Then the animat applies the same procedure with probability ϵ_X to the external action part. The overall strategy is a generalization of the "*compound*" action selection strategy that was introduced in (Lanzi 1998b).
We apply an animat with one bit of internal memory to Woods100 for values of ϵ_M and ϵ_X ranging from 0 to 1

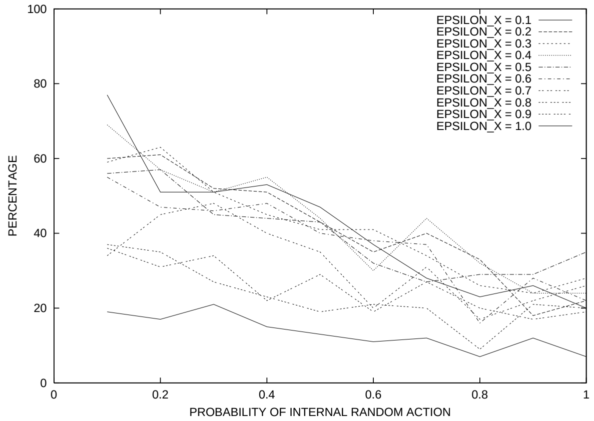

Figure 10: Number of *optimal* solutions for different internal exploration rates (ϵ_M) and a fixed external exploration rate (ϵ_X). The statistics are a measure over 100 runs of 50000 trials each.

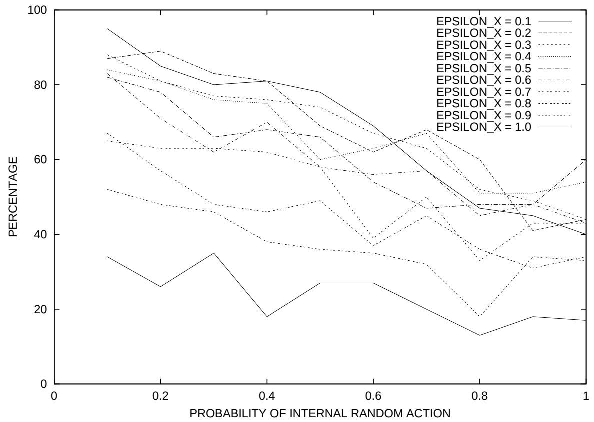

Figure 11: Number of *correct* solutions for different internal exploration rates (ϵ_M) and a fixed external exploration rate (ϵ_X). The statistics are a measure over 100 runs of 50000 trials each.

with step 0.1. For each ϵ_X-ϵ_M pair we made one hundred runs each one of 50000 trials. The number of *optimal* solutions developed for each ϵ_X-ϵ_M pair is showed in Figure 10; each line represents the number of optimal solutions developed by the animat with respect to a fixed external exploration rates (ϵ_X). As can be noticed for a limited external exploration rate ($\epsilon_X \leq .7$) the number of optimal solutions mainly depends on the internal exploration rate ϵ_M: the lower the ϵ_M the greater the number of optimal solutions developed. However, when the animat explores the environment *too much* ($\epsilon_X > .7$) then the number of optimal solutions drops dramatically. Similar results are showed in Figure 11 with respect to the number of *correct* solutions.

Our results, coherently with those presented in (Lanzi 1998b), suggest that the development of optimal strategies for aliased situations relies heavily on the balance between internal and external exploration. In general the animat has to explore the internal memory less than the environment; however, it must not explore the environment too much otherwise it is still unable to develop satisfactory behaviors in most of the cases.

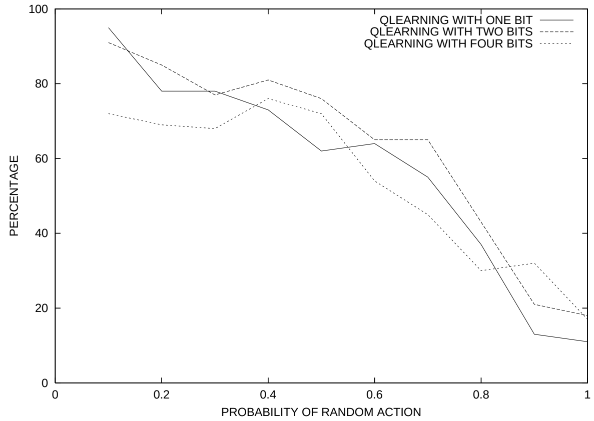

Figure 12: Number of *optimal* solutions developed with 1, 2, and 4 bits and different values of ϵ. The statistic is a measure over 100 runs of 50000 trials each.

Redundancy. Lanzi and Wilson (2000) also showed that a redundant number of internal memory bits can improve the learning capabilities of animats in perceptually aliased situations. They argued that more bits of internal memory means more optimal solutions and more chances that the animat can learn one of them.

We apply an animat with one, two, and four bits to Woods100 for one hundred runs. Figure 12 compares the number of *optimal* solutions developed by the three versions. For higher degrees of exploration (i.e., greater values of ϵ) there is a slight increase in the number of optimal solution developed. For small exploration rates (i.e., small values of ϵ) the animats with two and four bits develop fewer optimal solutions than that with only one bit. This results is easily explained: with more bits the animat has more solutions to explore; accordingly if the exploration rate is limited the animat may be unable to find a complete optimal solution. Therefore there is an *obvious* trade-off between the number of bits and the exploration rate: the more the bits the more the optimal solutions, but also, the more the exploration required to find an optimal solution. More interesting are the results concerning the number of *correct* solutions that the animats with one, two, and four bits can develop (see Figure 13). As the number of bits of internal memory increases the number of *correct* solutions developed increases: with four bits the animat finds a *correct* solution in more or less the 90% of the runs.

These results confirm what was found in (Lanzi and Wilson 2000): a redundant number of memory bits improves the learning capabilities of animats in perceptually aliased situations. However, one issue remains, that of scalability. Lanzi and Wilson (2000) showed that even if a very large number of memory bits was employed the learning problem remained computationally tractable. In our case, this will hardly hold anymore since the tabular representation grows exponentially in the number of memory bits. Again, this suggests that non-tabular reinforcement learning techniques (e.g., neural network or learning classifier systems) may be more profitable when using internal memory.

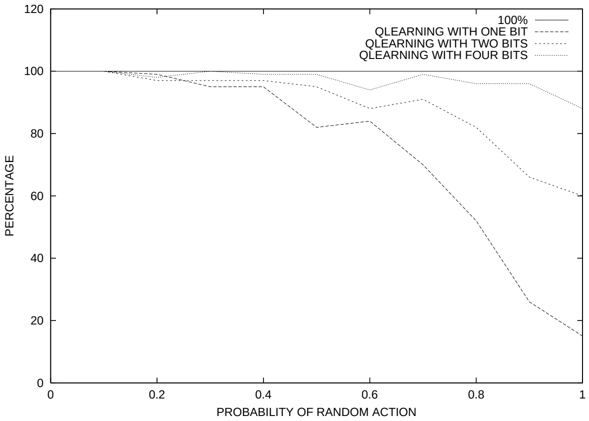

Figure 13: Number of *correct* solutions developed with 1, 2, and 4 bits and different values of ϵ. The statistic is a measure over 100 runs of 50000 trials each.

Eligibility Traces. Eligibility traces were first introduced by Sutton (1988); they can be viewed as a temporary (numerical) record, a "*memory*", of the occurance of an event. Thus they appear to be well-suited for perceptual aliasing. Recently, Loch and Singh (1999) studied the use of eligibility traces in non-Markov problems within *memoryless* approaches. Peshkin et al. (1999) used eligibility traces with SARSA and memory reporting interesting results. We add eligibility traces to Q-learning with internal memory following the approach proposed by Watkins (1989). We apply Q(λ)-learning with one bit of memory to Woods100 with different values of ϵ for one hundred runs. The number of optimal solutions developed is depicted in Figure 14; as can be noticed the use of eligibility traces *significantly* improves the learning capabilities of the animat. In fact, if we compare the number the *optimal* solutions with that of *correct* solutions in Figure 15 we see that almost all the correct solutions developed are also optimal. On the other hand we also note that still the exploration rate is a very important parameter: if ϵ is greater than 0.6 the number of optimal and correct solutions developed dramatically decreases. These results somehow summarize the results presented by different authors. In fact eligibility traces are effective in perceptually aliased problems (Peshkin et al. 1999) but still the balance with exploration is an important issue (Lanzi 1998b).

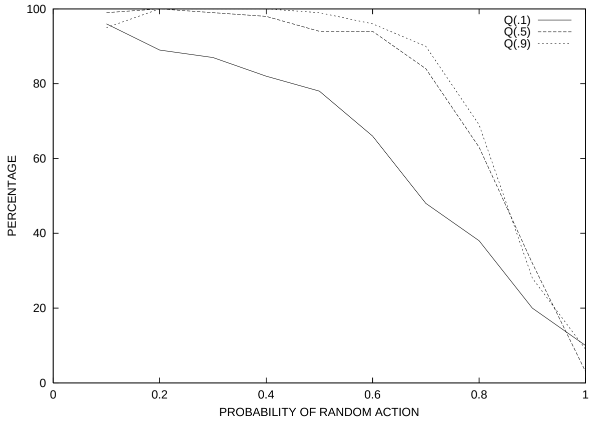

Figure 14: Number of *optimal* solutions for Q(λ) for λ equal to 0.1, 0.5, and 0.9. The statistics are a measure over 100 runs of 50000 trials each.

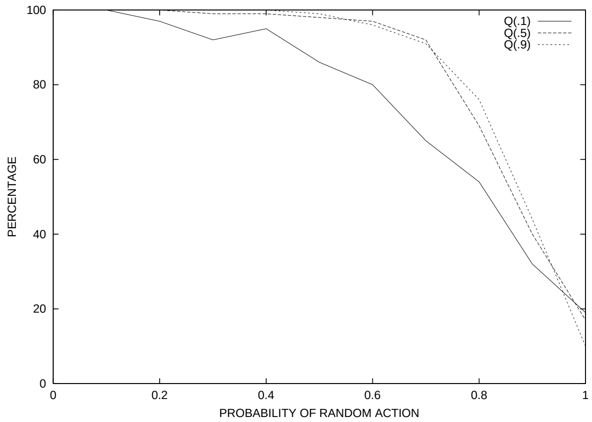

Figure 15: Number of *correct* solutions for Q(λ) for λ equal to 0.1, 0.5, and 0.9. The statistics are a measure over 100 runs of 50000 trials each.

12. Conclusions

We have studied the effectiveness of internal memory within the reinforcement learning framework. We have considered the most known reinforcement learning algorithm, Q-learning, and added internal memory. We have shown that internal memory can solve perceptual aliasing but in certain cases it might also introduce a new type of aliasing "*on the payoffs*" which in principal can prevent the development of optimal or just satisfactory solutions. We have shown that to achieve satisfactory performance the animat *should not* learn a complete mapping from sensations-actions pairs to payoffs. Accordingly, an adequate exploration strategy can be effective in limiting the effects of the *aliasing on the payoffs*. However this solution does not guarantee the convergence to satisfactory performance in general.

Then we have considered three extensions that were previously introduced to improve the performance of reinforcement learning algorithms with internal memory. We have shown that, as suggested by Lanzi (1998a), the separation between the exploration of internal memory and external environment helps in defining and tuning an adequate exploration strategy so as to improve the learning capabilities of the agent. Furthermore, as showed by Lanzi and Wilson (2000), redundancy on the internal memory increases the number of good strategies that the animat can developed improving its learning performance. Finally, as suggested by Peshkin, Meuleau, and Kaelbling (1999), eligibility traces play an important role in improving the number of optimal solutions that the animat can develop, although their performance still depends on the rate of exploration of internal memory.

Overall with this paper we tried to identify some novel and relevant issues concerning the use of internal memory within the reinforcement learning framework while also presenting an initial step toward an unified view of some results that were previously reported with respect to different frameworks.

Acknowledgements

I wish to thank Marco Colombetti for the many discussion regarding internal memory in reinforcement learning and for his invaluable support to my researches.

Stewart Wilson for the many discussions, his invaluable effort to improve my English, and because he always prods me in my researches: part of the work presented here was inspired under his "*pressure*."

References

Cliff, D. and S. Ross (1994). Adding Temporary Memory to ZCS. *Adaptive Behavior 3*(2), 101–150.

Holland, J. H. (1986). Escaping Brittleness: The possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In Mitchell, Michalski, and Carbonell (Eds.), *Machine learning, an artificial intelligence approach. Volume II*, Chapter 20, pp. 593–623. Morgan Kaufmann.

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

Lanzi, P. L. (1998b). An analysis of the memory mechanism of XCSM. pp. 643–651. Morgan Kaufmann: San Francisco, CA.

Lanzi, P. L. and S. W. Wilson (2000). Toward optimal classifier system performance in non-Markov environments. *Evolutionary Computation*. to appear.

Loch, J. and S. Singh (1999). Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In *Machine Learning: Proceedings of the Fifteenth International Conference on Machine Learning*, San Francisco, CA. Morgan Kaufmann.

McCallum, A. K. (1995, December). *Reinforcement Learning with Selective Perception and Hidden State*. Ph. D. thesis, University of Rochester, Rochester, NY.

McCallum, R. A. (1996). Hidden state and reinforcement learning with instance-based state identification. *IEEE Transations on Systems, Man and Cybernetics - Part B (Special issue on Learning Autonomous Robots) 26*(3).

Peng, J. and R. Williams (1996). Incremental Multistep Q-learning. *Machine Learning 22*, 283–290.

Peshkin, L., N. Meuleau, and L. P. Kaelbling (1999). Learning policies with external memory. In I. Bratko and S. Dzeroski (Eds.), *Machine Learning: Proceedings of the Sixteenth International Conference*, San Francisco, CA, pp. 307–314. Morgan Kaufmann.

Stolzmann, W. (2000). An Introduction to Anticipatory Classifier Systems. Volume 1813 of *LNAI*, Berlin, pp. 175–194. Springer-Verlag. (this volume).

Sutton, R. and A. Barto (1998). *Reinforcement Learning – An Introduction*. MIT Press.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. In *Machine Learning 3*, pp. 9–44. Boston: Kluwer.

Watkins, C. (1989). *Learning from delayed reward*. Ph. D. thesis.

Wiering, M. and J. Schmidhuber (1998a). Fast On-Line Q(λ). *Machine Learning 33*, 105–115.

Wiering, M. and J. Schmidhuber (1998b). Hq-learning. *Adaptive Behavior 6*(2), 219–246.

Wilson, S. W. (1985, July). Knowledge Growth in an Artificial Animal. pp. 16–23. Lawrence Erlbaum Associates: Pittsburgh, PA.

Wilson, S. W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation 2*(1), 1–18.