# Solving POMDPs using selected past events.

**Alain Dutech**[1]

**Abstract.** We present new algorithms for solving Partially Observed Markov Decision Processes. These algorithms are build on theoretical results showing that if one can find an observable with required properties, it is possible to build an extension of the state space using past events which defines a Markov Decision Process equivalent to the original problem. Thus, solving POMDPs, which is a very hard task, is seen as solving a MPD, where numerous existing algorithms can be successfully used. Our first algorithm uses reinforcement learning to solve POMDPs when the evolution model is unknown, whereas our second algorithm, more efficient, can only be applied when the model is known.

## 1 INTRODUCTION

We are interested in situated agents which have to plan actions in uncertain and unknown environments. Partially Observed Markov Decision Processes are good models of such problems, but can also be used in many other applications, see [1]. In this formalism, the agent interacts with its environment and tries to chose actions that maximize its cumulative reward. But solving POMDPs is a much more difficult task than solving Markov Decision Processes (MDPs), see [2], mainly because a given observation can be associated to many different states of the underlying MDP. *Belief states*, as proposed by Aström in [3], have successfully been used in [4] but are limited to problems with very few states where the model of the POMDPs is known, even using approximations like in [5]. Others, like [6], try to use perceptual actions to gain non-ambiguous knowledge about the current state.

Lin (see [7]) was one of the first who tried using the past to build non-markovian policies for POMPD with recurrent neural networks. His succes was limited, mainly because its use of past information was undirected. Our idea, like in [8], is to use the past of the process in a selective way so as to build a minimal extended state space on which classical algorithms for MDPs can be run.

In Section 2 of this paper we introduce the basics of POMDPs and of trajectories of past events which we use thereafter. The following section gives theoretical results about the required properties of a good extended state space. We build on these theoretical results to propose two algorithms for solving POMDPs. The first one can be used when the model is not known (see Section 4), the other when the model is known (Section 5). We end this paper by giving experimental results and a short conclusion.

## 2 PRELIMINARIES AND NOTATIONS

We begin by giving a definition of Markov Decision Processes (MDP) and of the more general Partially Observed Markov Decision Processes (POMDP) before introducing the notion of Observation-Action Trajectories (OAT).

For a finite state space $\mathcal{S}$ and a finite action space $\mathcal{A}$, a **Markov Decision Process** is defined by a **transition function** $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1]$ and a **reward function** $\mathcal{R} : \mathcal{S} \times \mathcal{A} \longrightarrow \Re$. $\mathcal{T}(s, a, s')$ gives the probability of reaching state $s$ from state $s'$ after executing action $a$. $\mathcal{R}(s, a)$ is the expected reward obtained while executing action $a$ in state $s$.

A **policy** is a mapping $\pi : \mathcal{S} \longrightarrow \mathcal{A}$. Our goal is generaly to find a policy maximizing a given function of the reward, as, for exemple, $\sum_{t=0}^{\infty} \gamma^t R_t$ where $R_t$ is the reward received at time $t$. In that case, $\gamma$ is positive real number smaller than 1.

The **model** of a MDP consists of both $\mathcal{T}$ and $\mathcal{R}$. The problem of finding an optimal policy $\pi^*$, which in the case of a MDP is deterministic and markovian, can be solved with or without knowledge of that model, as shown in [9], [10].

We are interested in a more general and complex type of problems, which are formulated as POMDP. A POMDP is a MDP where the state of the problem is not perfectly known but only observed through an observation function $\mathcal{F} : \mathcal{S} \longrightarrow \mathcal{O}$, where $\mathcal{O}$ is a finite set of observations. We must point out that we do not consider here the more general case where the observation function can be stochastic but limit ourselves to the deterministic case, which is still quite complex. Thus, $o = \mathcal{F}(s)$ is the observation associated with state $s$ and the difficulty lies in the fact that more than one state can be associated to a single observation.

The problem is now to find a policy which depends only of the observations, we call this an **adapted policy**. As explained by Singh and Jaakkola in [2], such an optimal adapted policy needs not be deterministic, nor even markovian. Our approach relies on what we have called Observation-Action Trajectories.

Let us first give the definition of an **State-Action Trajectory** (SAT). For any given state $S_t$, it is the possibly infinite sequence $\sigma = (S_t, A_{t-1}, S_{t-1}, \ldots, A_{t-k}, S_{t-k}, \ldots)$ composed of past events. $\Sigma$ is the set of all Action-State Trajectories. We define the set $\Omega$ of **Observation-Action Trajectories** (OAT) in much the same way by saying that an OAT is an infinite sequence $(O_t, A_{t-1}, O_{t-1}, \ldots, A_{t-k}, O_{t-k}, \ldots)$. For an SAT $\sigma$ there is a unique OAT $\omega$ such that $\omega = \mathcal{F}(\sigma) = (\mathcal{F}(S_t), A_{t-1}, \mathcal{F}(S_{t-1}), \ldots)$.

Finally, we will use finite $n$-**order Trajectories** which are trajectories truncated at time $t - n$, i.e.$(s_0, a_1, s_1, a_2, s_2)$ is a 2-order Trajectory. For a given policy $\pi$, one can compute the probability that a $n$-order SAT was followed as $P\{\sigma\} = P\{S_{t-n}\} \prod_{k=1}^{n} (\sum_{a \in \mathcal{A}} \mathcal{T}(S_{t-k}, a, S_{t-k+1}) \times P\{\pi(S_{t-k}) = a\})$. We call the second term of this product the likelyhood of the trajectory given $S_{t-n}$. The same computation can be done for an action-observation trajectory.

[1] LORIA, Campus Scientifique, BP239, 54506 Vandoeuvre les Nancy France, email: dutech@loria.fr

## 3 THE MAIN THEOREM BEHIND OUR SOLUTION

### 3.1 Overview

The main idea of our method is to use the recent past of the observed process to find the appropriate action to be executed. We will work on an extended state space composed of $n$-order Action-Observation Trajectories and try to solve the original POMDP by using MDP resolution algorithms on this extended space. To put it in another way, we look for a non-markovian adapted policy for the POMDP. Yet it is very unpractical, not to say nearly impossible, to simply consider an extended space made of all the different $n$-order Action-Observation Trajectories due to the related combinatorial explosion. Such a set would grow exponentially large with $n$.

But we have noted that, in most cases, we can restrict ourselves to a smaller extended space on which it is still possible to derive an optimal policy. Intuitively, the reason for this is that the knowledge of the last $n$ observations of the process is not always required to have enough information on the actual state of the process. This is illustrated in Figure 1 where one can say with certainty that the actual state of the process is $s2$ if one observes *blue* then *red*, no more information on the past is needed. But when *red* is observed two times in a row, it is not enough to know the state of the process and one must use 2-order trajectories.

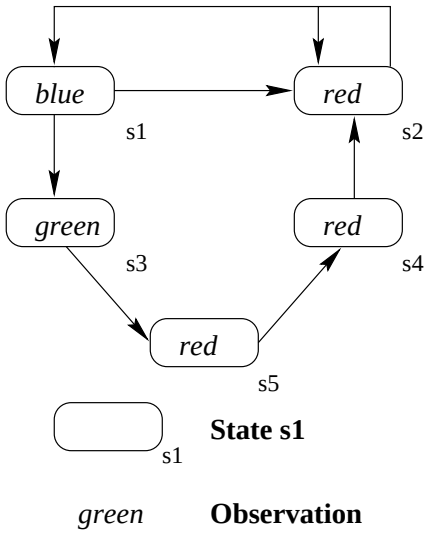


**Figure 1. An example of POMDP.** In this example, we have 5 states, 3 observations and only one action $a$. An exhaustive observable would then be (*blue*), (*green*), (*red*,$a$,*blue*), (*red*,$a$,*green*), (*red*,$a$,*red*,$a$,*green*), (*red*,$a$,*red*,$a$,*red*)

So, our goal is to build a minimal extended state space with finite order OAT, all these trajectories can be of different order. For each possible actual finite or infinite SAT of the process one should be able to associate one unique element of the extended space, and this element **should give us enough information so as to choose an optimal action**, (knowing exactly the state of the process is an even stronger requirement). We call an extended state space meeting this requirement an exhaustive observable $\mathcal{H}$ and we write $proj_{\mathcal{H}} : \Sigma \longrightarrow \mathcal{H}$ the function which associates a unique element of $\mathcal{H}$ to any trajectory of the process. We now give more formal definitions of these notions.

### 3.2 Formal definitions and Theroem

An **observable** is a set of finite order Action-Observation Trajectories.

An observable $\mathcal{H}$ is **complete** iff for each SAT $\sigma$, there exists a unique element $\omega$ of $\mathcal{H}$ such that $\omega = proj_{\mathcal{H}}(\sigma)$.

An complete observable $\mathcal{H}$ is **exhaustive** if it has the following property [2] : $P\{S_{t+1}|a, \Sigma_t\} = P\{S_{t+1}|a, proj_{\mathcal{H}}(\Sigma_t)\}$ where $\Sigma_t$ is the potentially infinite Action-State Trajectory of the process.

**Theorem 1** *If $\mathcal{H}$ is an exhaustive observable for a given POMDP $\lambda$, then the extended decision problem defined by $(\tilde{S} = proj_{\mathcal{H}}(\Sigma_t))_t$ is a Markov Decision Problem $\tilde{\lambda}$.*

**Theorem 2** *The optimal policy $\tilde{\pi}$ for the extended Markov Decision Problem $\tilde{\lambda}$ defines also an optimal* **adapted** *policy for the original POMDP $\lambda$ in the sens that the reward received by applying $\tilde{\pi}$ is the same that the one which would be received by applying the optimal non-adapted state policy $\pi^*$.*

The detailed proofs of the theorems can be found in [11].

The consequence of all this is that if we can find a finite exhaustive observable for a POMDP, solving the POMDP is the same as finding an optimal policy for the extended Markov Decision Process defined by the observable. And methods for solving MDPs are numerous. One last notice: to derive these theorems, one must assume that the transition function is deterministic. But, experiments have shown that algorithms derived from these theorems work also with non-deterministic transition function.

## 4 UNKNOWN MODEL

In this section we present briefly the results we obtained when the model of the POMDP is not known, more detailed results can be found in [11]. We decided to present this part because the general ideas to solve the problem when the model is known are roughly the same than when the model is not known. In fact, the methods of this Section could also be used when the model is known, as stressed in Section 5.4.

The algorithm we have developped is largely based on a widely used reinforcement algorithm : Q-Learning, proposed by Watkins in [10]. Q-Learning can solve MDPs by iterative asynchronous approximations of a function $Q(s, a)$ which gives an estimation of the future expected reward for each (state,action) pair. An optimal policy is then $\pi^*(s) = argmax_{a \in \mathcal{A}} Q(s, a)$. When used on POMDP with (observation,action) pairs, Q-Learning cannot converge unless one uses a fixed policy for generating (observation,action) pairs (see [2]). The policy thus derived is sub-optimal. Of greater importance, the speed of convergence of the $Q(o, a)$ function is affected by the ambiguity of the observation $o$: if $o$ is ambiguous, i.e. is associated to more than one state, the convergence of $Q$ is slower than for non-ambiguous elements. Thus, by looking at the variations of the $Q$ function, *we now have a way to detect the most ambiguous elements of an observable...*

We intialize our algorithm with an observable made from all the possible 0-order OAT (i.e. the observations). Our algorithm is described below.

[2] An observable must also have other properties over the observation function and the reward distribution as well to be exhaustive, to assure that the reward distribution is coherent with the observable. Such properties are generally met and we omit them here because of lack of space. The reader should refer to [11] for more detailed results.

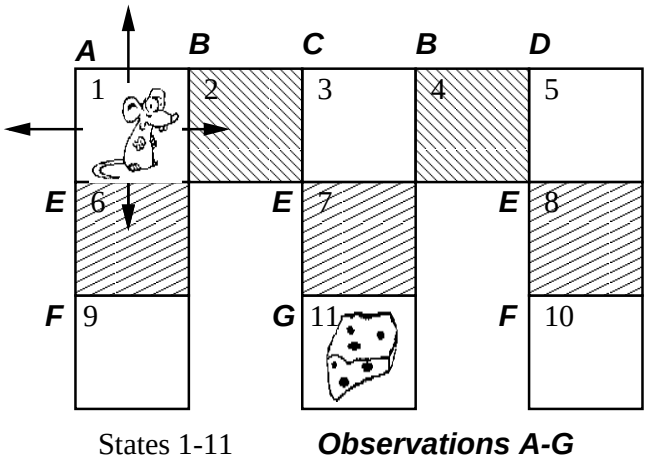


**Figure 2. The Cheese Maze.** In this example, the mouse must find the cheese. There are 11 states and 7 observations, some (B and E) beeing ambiguous.

- **Step 1: Growing the observable** By running a version of Q-Learning with a fixed stochastic exploration policy on this observable, we are able to detect ambigous elements of $\mathcal{H}$, as stated above. These elements form the set $\mathcal{M}$. Each element $\omega$ of $\mathcal{M}$ is the base of $|\mathcal{A}| \times |\mathcal{O}|$ higher order OATs looking like $(\omega, a, o)$, which are added to $\mathcal{H}$.
- **Step 2: Best current policy** Once a stable observable is reached, using Q-Learning on the extended state space defined by the observable, we find the best policy for the current extended process. The reward received by applying this policy gives us an estimation of the performances of the current observable. By theorems 1 and 2, this policy is optimal for the POMDP when we have an exhaustive observable.
- **Step 3:Iterate if not exhaustif** In fact, we cannot detect exhaustive observables with our method. We can only detect the most ambiguous elements of the observable. So, we stop our algorithm when the difference in performances of two successive observables are negligeable. Otherwise we go back to Step 1.

This agorithm has been tested on different problems, like the well known Cheese Maze depicted in Figure 2. Figure 3 shows the results for this problem, and one can easily see the alternating *Growing* and *Evaluating* steps of the algorithm. Our algorithm was found to be robust but suffers from the large number of iterations needed to reach an optimal policy. It is this limitation we try to work on in the following Section.

# 5 KNOWN MODEL

In this section, we suppose that the model of the POMDP is known and that the transition function is deterministic. It can seem strange to try to solve a POMDP when the model is known as one could more easily solve the underlying MDP. But, in the case of a situated agent which can only rely on the observations given to him, the optimal policy based on *states* is of no use. This is why finding an adapted policy (i.e. based on the *observations*) is so important.

Our goal is to use the information of the model to explicitely compute an exhaustive observable of the POMDP. This observable defines an extended MDP which we can then solve in fewer iterations than in Section 4 using algorithms like *value iteration* or *policy iteration* (see [9]).

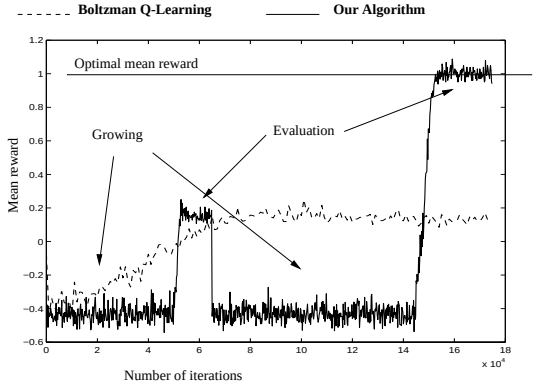


**Figure 3. Result on the Cheese Maze.** We have tested our algorithm against an algorithm giving only markovian policies. After tho epoch of *Growing* and *Evaluating*, our algorithm converges to the optimal solution, which is non-markovian (in fact 2-order markovian).

As developped in this section, we will see that it is indeed impossible to find an exhaustive observable. Thus, in order to solve correctly the POMDP we chose to rely on quasi-exhaustive observables and Q-Learning, as explained in Section 5.4.

## 5.1 Exhaustivity is a dream

When trying to find a proper algorithm for computing exhaustive observable we eventually found out that, in the general case, there are no finite order exhaustive observables. Figure 4 shows this on a very simple example where, by moving alternatively Up and Down, an agent generates an infinite SAT which cannot be associated to any finite order non-ambiguous OAT. In that case, knowledge of all the past of the process is required to know the state of the underlying MDP.

This example shows that two different states cannot be told appart, even with a total knowledge of the past. If the reward and transition functions of these states are different, a finite-order exhaustive observable does not exists. And this is the case for most of the problems we try to solve.

One can ask how it is, then, that optimal solutions and exhaustive observables where found in Section 4, when no model was known. The answer is that, with some constraints on the policy, it is possible to find such observables. Without a model, our algorithm incrementally constructed observables which where exhaustive when applying an optimal policy. By itself, the algorithm payed few attention to ambiguous but very sub-optimal Action-Observation Trajectories which were then ruled out.

But here, when trying to compute the observable without any idea on what the optimal policy could be, it is impossible to put some ambiguous trajectories aside. So, we decided to use weaker observables which are only quasi-exhaustive.

## 5.2 Quasi-exhaustive observables

The idea is to build an observable where each element is either a non-ambiguous trajectory or a ambiguous trajectory but which is very unlikely to be followed by the agent. We have seen in Section 2 that this last notion is linked to what we called the likelyhood of a trajectory, furthermore it is rather easy to compute this likelyhood. The complexity of the calculus is in $O(n)$ if $n$ is the order of the trajectory. To know if a trajectory is ambiguous, we evaluate the number

of possible final states compatible with the trajectory. If this number is greater than 1, the trajectory is ambiguous.

We can now present our algorithm for building a quasi-exhaustive observable $\mathcal{H}$.

- **Step 1: Initialization** Set $\mathcal{H}$ to $\mathcal{O}$, the observation space.
- **Step 2: Precalculation** For every element of $\mathcal{H}$, compute the number of associated final states, complexity in $O(|\mathcal{S}|)$. If this number is greater than 1, an element is ambiguous. Likewise, compute the likelyhood of every element, complexity in $O(|\mathcal{O}| \times |\mathcal{S}|)$.
- **Step 3: Growing** For every ambiguous element of $\mathcal{H}$ with a likelyhood above a given threshold, add $|\mathcal{O}| \times |\mathcal{A}|$ new trajectories to $\mathcal{H}$. If no element are added, **Stop**.
- **Step 4: Update** For each new element, see if it is ambiguous, complexity in $O(|\mathcal{A}| \times |\mathcal{S}|)$. For each new element, compute its likelyhood, complexity at most in $O(|\mathcal{A}| \times |\mathcal{S}| \times n)$ where $n$ is the order of the element; this computation can be made less complexe by using previous results. Go to Step 3.

## 5.3 Loopholes

It is possible to derive a new transition function and a new reward function for the extended state space build on a quasi-exhaustive observable. But, as we will explain, it is not very useful as one can not directly apply model based algorithm on quasi-exhaustive observable. Indeed, the transition function is needed when we want to use a model based algorithm like *policy iteration* to solve the MDP. But, when we keep ambiguous extended states, even if they seem unlikely to be used, we create *loopholes* if the model.

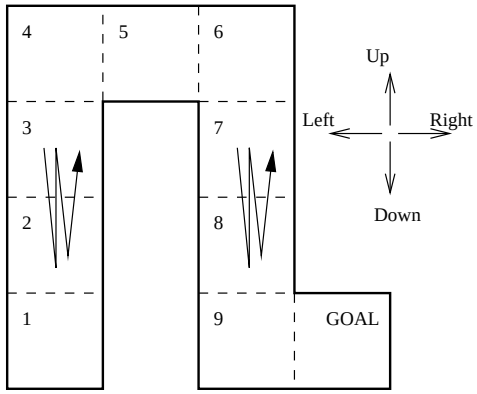


**Figure 4. Simple environment.** In this simple environment, there are 10 states and 4 actions. The agent observes only the position of the walls for each cell, so cells 2 and 8 look the same to it. Furthermore, even using the two inifinite trajectories depicted, states 3 and 7 cannot be taken apart.

Figure 5 gives a representation of the model associated with the problem depicted in Figure 4 when whe use raw observations as extended space. Some observation are ambiguous, and the agent cannot make the difference between state 2 and state 8. Then, starting in state 1, the optimal policy seems to go Up and then Down in extended state $2/8$.

This simple example is an image of what happen in complex environment when we try to use ambiguous elements in the extended space. Event though these elements seem unlikely to be used, model base algorithms will have a tendancy to abuse them if they are part of a loophole which appears optimal, as in the exemple of Figure 5. As a consequence, model-based algorithms cannot be used with quasi-exhaustive algorithms.

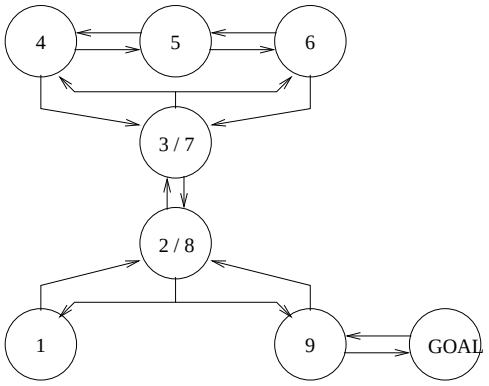


**Figure 5. Loophole in the model.** If the agent uses ambiguous observations, it creates virtual transition in the model. Here, for example, states 2 and 8 are aggregated which creates a shortcut from 1 to 9, shortcut which will be used by a model-based algorithm like policy iteration.

## 5.4 Actual Solution

Apart from preventing the agent from taking actions which could lead him on ambiguous observation-action trajectories, actually acting in the environment is the only way for the agent not to fall in the loopholes induced by ambiguous trajectories. If the agent takes an action which, it believes, leads him to an "interesting" ambiguous situation which is in fact a loophole, the environment itself will prevent the agent from taking advantage of this false belief. Going back to the exemple set in Figure 5, coming from 1 into $2/8$, the agent will never be able to go directly to state 9 as a model could lead him to believe.

As a consequence, our algorithm for solving POMDP with a known model is the following.

- **Step 1: Quasi-Exhaustive Observable** Following the algorithm of Section 5.2, a quasi-exhaustive observable is built. The threshold and the maximum order of elements in the observable govern its final size.
- **Step 2: Q-Learning** It is a simple Q-Learning where the balance between exploration and exploitation is done using a Boltzman distribution of probability for choosing the next action (see [12]).

**Table 1.** Result for the first environment

| Max. Order | Nb | Goal | Nb. Iter. | % Optim. |
|---|---|---|---|---|
| states | 14 | 14 | 800 | 100 |
| 0-order | 9 | 14 | 800 | 0 |
| 1-order | 17 | 14 | 5000 | 66 |
| 2-order | 23 | 9 | 3000 | 100 |
| 2-order | 23 | 4 | 3000 | 100 |
| 2-order | 23 | 11 | 3000 | 100 |

We have tested our algorithm on two different environments. The first one is the environment of Figure 6, with 14 states and 9 observations. For this environment we tried 4 different locations for the *GOAL*. When it reaches the goal, the agent is randomly taken to a new starting position. The reward for the agent: +5.0 for the goal,

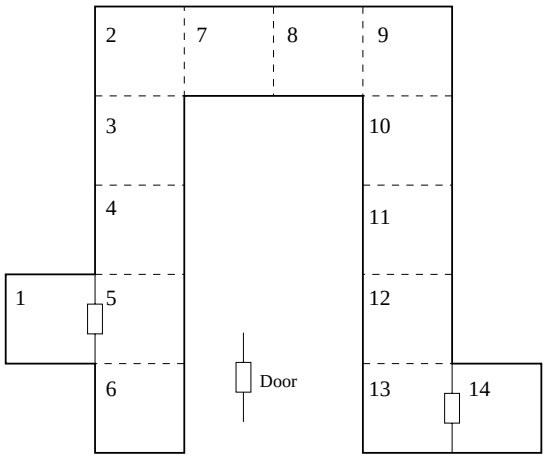


**Figure 6. First test environment.** The *goal* state is set for each simulation. See results in Table 1

-0.2 when moving and -15.0 for bumping into a wall. $\gamma$ has a value of $0.99$. Table 1 gives the results for this environment, in term of the maximum order for the observable, the actual number of elements in the extended state, the number of iterations for the Q-Learning algorithm and the percentage of optimal policy obtained. Table 2 gives the same information for our aglorithm tested on the environment shown in Figure 7, with 89 states and 24 observations.

For each environment we can find optimal policies with a very reasonnable number of iterations. The size of the observable stays small and ambiguous elements do not prevent us from reaching optimal solutions.

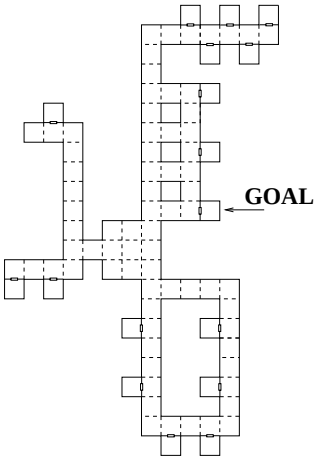


**Figure 7. Second test environment.**

## 6 CONCLUSION

This paper presented new algorithms for solving Partially Observed Markov Decision Processes by using a carefully crafted extended space. This extension relies on using past information to derive non-markovian policies, and combinatorial explosion is prevented by looking for minimal exhaustive observable. After presenting two theorems underlying our method, we described the algorithms that can

**Table 2.** Result for the second environment

| Max. Order | Nb | Nb. Iter. | % Optim. |
|---|---|---|---|
| states | 89 | 50000 | 100 |
| 0-order | 24 | 50000 | 0 |
| 1-order | 100 | 100000 | 0 |
| 2-order | 185 | 30000 | 100 |
| 2-order | 185 | 15000 | 58 |
| 2-order | 185 | 12000 | 20 |

be used, depending on the knowledge we have of the evolution model of the process.

This works open further interesting lines of research. Most important, the influence of noise in the model should be investigated. Beside, our algorithm takes only into account the $n$ last past events at most and we have ideas for using past events which occurred at non-fixed time step backward in time which we hope to develop in future works.

## REFERENCES

[1] Cassandra, A. *A survey of POMDPs applications.* Planning with Partially Observable Markov Decision Processes, American Association for A.I. Symposium, (1998).

[2] Singh, S., Jaakkola, T. & Jordan, M. Learning without state estimation in partially observable markovian decision processes. *Proceedings of the Eleventh International Conference on Machine Learning*, (1994).

[3] Aström, K. (1965). Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications 10*, 174-205.

[4] Littman, M. *The Witness algorithm: solving partially observable Markov decision processes.* Technical report CS-94-40, Computer Science Department, Brown University, Providence, Rhode Island, (1994).

[5] Cassandra, A. *Exact and approximate algorithms for partially observable Markov decision processes.* Ph.D. Thesis, Computer Science Department, Brown University, Providence, Rhode Island, (1998).

[6] Chapman, D. & Kaelbling, L. Input generalization in delayed reinforcement learning: an algorithm and performance comparisons. *Proceedings of the International Joint Conference in Artificial Intelligence.* Sydney,(1991).

[7] Lin, L-J. Self-improvement reactive agent based on reinforcement learning, planning and teaching. *Machine Learning, 8*, 293-321, (1992).

[8] McCallum, A. Learning to use selective attention and short-term memory in sequential tasks. *Proceedings of the Fourth International Conference on Simulating Adaptive Behavior*, MIT Press, Cambridge, Massachusetts, (1996).

[9] Puterman, M. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, Inc. (1994).

[10] Watkins, C. *Learning from delayed rewards*. Ph.D. Thesis, King's College, Cambridge, England, (1989).

[11] Dutech. A. *Apprentissage d'environnements: approches cognitives et comportementales.* Ph.D. Thesis, Artificial Intelligence Department, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, France, (1999). *In French*.

[12] Thrun, S. *Efficient exploration in reinforcement learning*. Technical report CMU-CS-92-102, Computer Science Department, Carnegie Mellon University, (1992).