Learning a Machine for the Decision in a Partially Observable Markov Universe

Frédéric Dambreville
Délégation Générale pour l'Armement, DGA/CTA/DT/GIP/PRO
16 Bis, Avenue Prieur de la Côte d'Or
F 94114, France
Email: Frederic.DAMBREVILLE@dga.defense.gouv.fr

Abstract— **In this paper, we are interested in optimal decisions in a partially observable Markov universe. Our viewpoint departs from the dynamic programming viewpoint: we are directly approximating an optimal strategic tree depending on the observation. This approximation is made by means of a parameterized probabilistic law. In this paper, a particular family of hidden Markov models, with input *and* output, is considered as a learning framework. A method for optimizing the parameters of these HMMs is proposed and applied. This optimization method is based on the cross-entropic principle.**

I. Introduction

There are different degrees of difficulty in planning and decision problems. In most problems, the planner have to start from a given state and terminate in a required final state. There are several transition rules, which condition the sequence of decision. For example, a robot may be required to move from room A, starting state, to room B, final state; its action could be *go forward*, *turn right* or *turn left*, and it cannot cross a wall; these are the conditions over the decision. A first degree in the difficulty is to find at least one solution for the planning. When the starting state is only partially known or the actions are not deterministic, the difficulty is quite enhanced: the planner has to take into account the various observations. Now, the problem becomes much more complex, when this planning is required to be optimal or sub-optimal. For example, find the shortest trajectory which moves the robot from room A to room B. There are again different degrees in the difficulty, depending if the problem is deterministic or not, depending on the model of the future observations. In the particular case of a Markovian problem with the full observation hypothesis, the dynamic programming principle could be efficiently applied (Markov Decision Process theory/MDP). This solution has been extended to the case of partial observation (Partially Observable Markov Decision Process/POMDP), but this solution is generally not practicable, since the dimension of the variables grows exponentially.

In this paper, we are interested in optimal planning with partial observation. A Markovian hypothesis is made. Our viewpoint departs from the POMDP and the dynamic programming viewpoint: we are directly approximating an optimal strategic tree depending on the observation. This approximation is made by means of a parameterized conditional probabilistic law, *ie.* actions depending on observations. The main problems are the choice of the parameterized laws family and the learning of the optimal parameters. A particular family of (hierarchical) hidden Markov models, with input *and* output, has been used for the learning framework. A method for optimizing the parameters of these HHMMs is proposed and applied. This optimization method is based on the cross-entropic principle[1]. The resulting parameterized law could be seen as a *Virtual Machine* which could generate suboptimal dynamic strategies for any drawing of the problem.

The next section introduces some formalism and gives a quick description of the MDP/POMDP problems. It is recalled that the Dynamic Programming method could solve these problems, but that this solution is intractable for an actual POMDP problem. It is then proposed a new sub-optimal planning method, based on the direct approximation of the optimal decision tree. The third section introduces the family of Hierarchical Hidden Markov Models. A particular sub-family of HHMMs is proposed as a candidate for the approximation of decision trees. The fourth section describes the method for optimizing the parameters of the HHMM, in order to approximate the optimal decision tree for the POMDP problem. The cross-entropy method is described and applied. The fifth section gives an example of application. The paper is then concluded.

II. MDP and POMDP

It is assumed that a subject is acting in a given world with a given purpose or mission. The goal is to optimize the accomplishment of this mission. In the next paragraphs, a Markovian modelling of this problem is considered.

a) The world: In MDP/POMDP, the world is considered as a Markov chain conditioned by the past actions of the subject. Assume that the world is characterized by a *state* variable z and that the action is described by a variable x. Assume that the time is starting from 1, and denote respectively z_t and x_t the world state and the action for the time t. The law of z with respect to the past actions x thus verifies the property of Markov:

$$P(z_t|z_{1:t-1}, x_{1:t-1}) = p(z_t|z_{t-1}, x_{t-1}) ,$$

Fig. 1. A Markovian World

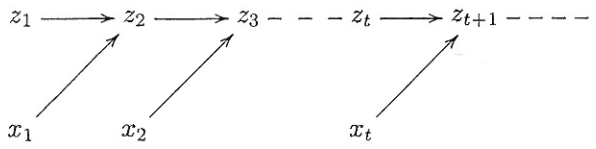
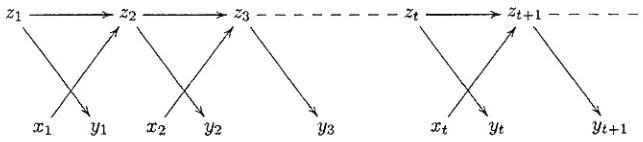


Fig. 2. A Hidden Markov Model (with control)



where the notation $z_{1:t}$ means $z_1, \ldots, z_t$, and the notation z_0, *ie.* before starting time, means $\emptyset$. This Markov chain is thus characterized by two functions: an initial law $p(z_1|\emptyset)$ and a transition law $p(z_t|z_{t-1}, x_{t-1})$ for $t \geq 2$. This distinction is often omited in this paper, but it is implied. The law of $z|x$ is represented graphically by the Bayesian Network of figure 1. In this description, the arrows indicate the dependency between variables. For example, $z_t \rightarrow z_{t+1} \leftarrow x_t$ means that the law of z_{t+1} is defined conditionally to the past variables x_t and z_t. Such Bayesian Network representations will be used several times in this paper for their pedagogic skill.

b) The observation: MDP and POMDP assume that the world is observed during the process.

In MDP, the world is fully observed; *the observation at time t is the world state z_t.*

In POMDP, the world is partially observed; *the observation at time t is denoted y_t.* The variable of observation y_t is only depending on the current world state z_t:

$$P(y_t|z_{1:t}, y_{1:t-1}, x_{1:t}) = p(y_t|z_t) \ .$$

The variables $y, z|x$ constitute a *Hidden Markov Model* (with control). This HMM is represented in figure 2.

c) Evaluation and optimal planning: The previous paragraphs have built a modelization of the world, of the actions and of the observations. We are now giving a characterization of the mission to be accomplished. *The mission is limited in time.* Let T be this maximum time. In the most generality, the mission is evaluated by a function $V(x_{1:T}, y_{1:T}, z_{1:T})$ defined on the trajectories of x, y, z ($V(x_{1:T}, z_{1:T})$ in the case of a MDP). Typically, the function V could be used for computing the time needed for the mission accomplishment. The purpose is to construct an optimal decision tree $x(obs)$ depending on the observation obs in order to maximize the *mean evaluation.* This is a dynamic optimization problem, since the actions depend on the previous observations. This problem is expressed differently for MDP and POMDP:

Fig. 3. MDP planning

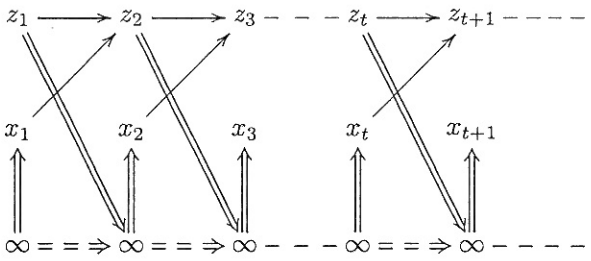
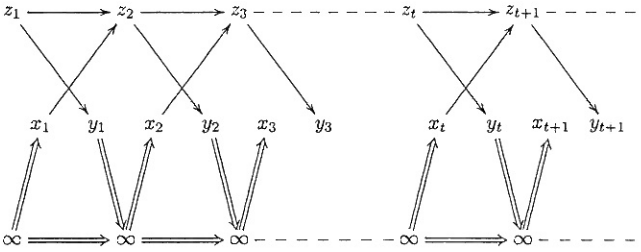


Fig. 4. POMDP planning



MDP. $obs_t = z_t$. Optimize $x_t(z_{1:t-1})|_{1 \leq t \leq T}$:

$$x_O \in \arg\max_x \sum_{z_{1:T}} V(x_t(z_{1:t-1})|_{1 \leq t \leq T}, z_{1:T}) \prod_{t=1}^{T} p(z_t|z_{t-1}, x_{t-1}(z_{1:t-2})) \ .$$

POMDP. $obs_t = y_t$. Optimize $x_t(y_{1:t-1})|_{1 \leq t \leq T}$:

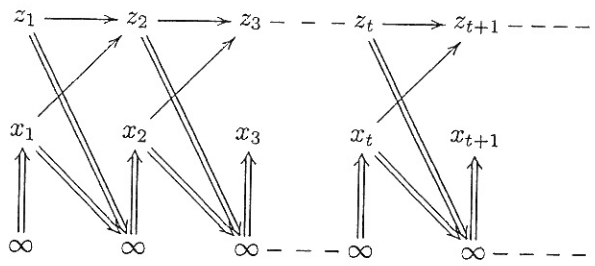
$$x_O \in \arg\max_x \sum_{y_{1:T}} \sum_{z_{1:T}} V(x_t(y_{1:t-1})|_{1 \leq t \leq T}, y_{1:T}, z_{1:T}) \prod_{t=1}^{T} p(y_t|z_t)\, p(z_t|z_{t-1}, x_{t-1}(y_{1:t-2})) \ . \quad (1)$$

These optimizations are schematized in figures 3 and 4. In these figures, the double arrows $\Rightarrow$ characterize the variables to be optimized. More precisely, these arrows describe the flow of information between the observations and the actions. *The cells denoted ∞ are making decisions and transmitting all the received and generated information* (including the actions). This architecture illustrates that planning with observation is an *infinite-memory* problem: the decision depends on the whole past observations. It is shown in section II-A, that MDP is in fact a finite memory problem: the dashed arrows of figure 3 may be removed resulting in the finite memory BN of figure 5 (memory of the past decision is maintained).

In fact, these theoretical constructions of the optimal decision tree are implemented in an actual open-loop planning process in order to optimize the actual mission:

1) Initialize the MDP/POMDP priors of the mission,
2) Compute x_O,
3) The first theoretical optimal decision $x_{O,1}$ is applied in the actual world,

Fig. 5. MDP planning with finite memory



4) Make actual observations and update the prior; in particular, the time is forwarded by 1,
5) Repeat from step 2) until accomplishment of the mission.

It appears that it is not necessary to build the whole optimal decision tree x_O for deciding the next actual move: the *root* $x_{O,1}$ of this tree is sufficient. A dynamic programming method may be applied for MDP, in order to compute $x_{O,1}$.

A. Dynamic Programming method

This presentation is extremely simplified. More detailed references are available[2][3][4]. The DP method works for an additive evaluation $V(x_{1:T}, z_{1:T}) = \sum_{t=1}^{T} V_t(x_t, z_t)$. In this simplified presentation, a final evaluation is assumed, *ie.* $V(x_{1:T}, z_{1:T}) = V_T(x_T, z_T)$, but the principle is the same. In the case of MDP, it has to be computed:

$$x_{O,1} \in \arg_{x_1} \max_{x} \sum_{z_{1:T}} V_T(x_T(z_{1:T-1}), z_T) \prod_{t=1}^{T} p(z_t|z_{t-1}, x_{t-1}(z_{1:t-2})) .$$

The solution is computed recursively (dynamic programming):

$$\begin{cases} W_T(x_T, z_T) = V_T(x_T, z_T) , \\ W_t(x_t, z_t) = \max_{x_{t+1}} \sum_{z_{t+1}} p(z_{t+1}|z_t, x_t) W_{t+1}(x_{t+1}, z_{t+1}) , \\ x_{O,1} \in \arg\max_{x_1} \sum_{z_1} p(z_1|\emptyset) W_1(x_1, z_1) . \end{cases}$$

The method thus relies on the computation of recursive functions defined over the variables x_t, z_t. The DP methods have been extended to POMDP problems, but this time, the variables for the recursive functions are probabilistic *beliefs* over z_t [3][4]. Probabilistic beliefs are continuous and are thus high-dimentional variables. For this reason, such methods are really difficult to apply practically.

This particular difficulty for solving POMDP problems is illustrated by the figures 3 and 4: MDPs are fundamentally more simple than POMDPs. Indeed, since the world state z is a Markov chain, the knowledge of z_t together with the last action x_t is sufficient to predict all the future. For this reason, the dashed arrows of figure 3 could be removed and replaced by arrows from the last action (figure 5): the problem is *finite memory*. Such simplifications are not possible for POMDP, since the knowledge y_t is not sufficient to predict the future of the Markov chain z.

B. Direct approximation of the decision tree

In an optimization problem like (1), the value to be optimized, x_O, is a *deterministic* object. In this precise case, x_O is a tree of decision, that is a function which maps to a decision x_t from any sequence of observation $y_{1:t-1}$. It is possible however to have a probabilistic viewpoint. Then the problem is equivalent to finding $\pi(x|y)$, a probabilistic law of actions conditionally to the *past* observations, which maximizes the mean evaluation:

$$V(\pi) = \sum_{x_{1:T}} \sum_{y_{1:T}} \sum_{z_{1:T}} \prod_{t=1}^{T} \pi(x_t|x_{1:t-1}, y_{1:t-1}) \times P(y_{1:T}, z_{1:T}|x_{1:T})\, V(x_{1:T}, y_{1:T}, z_{1:T}) ,$$

where:

$$P(y_{1:T}, z_{1:T}|x_{1:T}) = \prod_{t=1}^{T} p(y_t|z_t)\, p(z_t|z_{t-1}, x_{t-1}) .$$

If the solution is optimal, there will not be a great difference with the deterministic case: when the solution x_O is unique, the optimal law π_O is a dirac on x_O. But things change when approximating π. Now, why using a probability to approximate the optimal strategy? The main point is that probabilistic models seem more suitable for approximation. The second point is that we are sure to approximate continuously: indeed, $\pi \mapsto V(\pi)$ is continuous. There is a third point, which is more "philosophic". Considering the figure 4, it appears clearly that it is symetric: the arrows $\Rightarrow$ and $\rightarrow$ play a quite similar role. In the deterministic viewpoint, this is just a coincidence. But if we are considering probabilistic strategies, the arrows $\Rightarrow$ in the Bayesian Network of figure 4 are constituting a HMM controled by the observation, and with infinite memory. Of course, a natural approximation of a HMM with infinite memory is a HMM with finite memory! Then replacing the infinite-memory states ∞ by a finite memory variable m, a perfectly symetric problem is obtained in figure 6; perhaps some kind of duality relation. Well, aestheticism is not the unique interest of such approximation. The most interesting point is that there are many methods which applies on HMM. Particularly for learning and optimizing parameters. Then, the approach developped in this paper could be decomposed in two point:

- Define a family of parameterized HMMs,
- Optimize the parameters of the HMM in order to maximize the mean evaluation.

The first point is considered in the next section, where a subclass of hierarchical HMM is defined. The second point is considered in the following section, which explains a cross-entropic method for optimizing the parameters.

Fig. 6. Finite-memory planning approximation

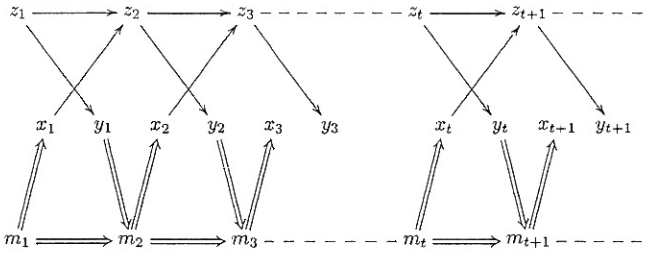
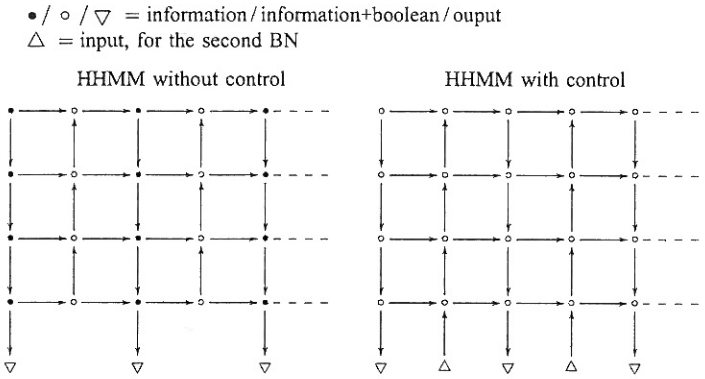


Fig. 7. Modeling of a Hierarchical HMM



III. Hierarchical Hidden Markov Model.

A. Generality

This section gives some general references about HHMMs *without control*. The author is not informed of works about controled HHMM. Thus, the model of controled HHMM defined later is not related to any existing work, the author knows. But it is interesting to introduce this section with some results about uncontroled HHMM.

A hierarchical hidden Markov model (HHMM) may be defined as a HMM which output is either a hierarchical HMM or an actual output. A HHMM could also be considered as a hierarchy of *stochastic* processes calling sub-processes. It is noteworthy that the length of a sub-process call depends on the sub-process law. This length is variable.

Simple HMM have been variously applied, for example in speech or hand-writing recognition, in robotic, etc. However there are still few applications of hierarchical HMM, although it is probable that HHMM should allow a more abstract representation of the processes.

In [5], *Fine, Singer and Tishby* applied HHMM to *identify combinations of letters in handwriting*. The main difficulty was to derive appropriate Viterbi and Baum-Welch algorithms. The Complexity of *Fine and al* methods was about T^3Q^Λ, where T is the length of the samples, Λ is the number of *levels* of the HHMM tree and Q is the number of state per levels of the HHMM. Such required computation times are obviously difficult to apply; for most samples T is too big.

In their work [6], *Murphy and Paskin* have shown how HHMM could be interpreted as a particular $2-$dimension dynamic Bayesian Network sized $\Lambda \times T$, and derived algorithms of complexity TQ^{2D}. This is much better, but remains exponential with Λ. The number of levels is thus strongly limited. Although *Murphy and Paskin* did not express this property explicitely, it is easy to show that their Bayesian Network has the Markov property both in time and in level. In fact, a hierarchical HMM could be interpreted by a Bayesian Network as described in the first picture of figure 7. A hierarchical HMM is thus clearly a HMM with vectorial states. However, these states should contains some boolean components which are used to define the sub-process call and return[6]. Moreover, there is a downward *and upward* exchange of information between the levels, in a hierarchical HMM.

The next section introduces a model of controled HHMM, which is (freely) inspired from the first BN presented in figure 7.

B. Model definition

A natural extension of the first BN of 7 is obtained by adding an input door to each upward column of the BN (then completing it), as described in the second picture of 7. It would be obtained an alternance of output and input, which exactly matches to our problem. This is probably a very good model. But for historical reason in our work, another model, similar, has been chosen...

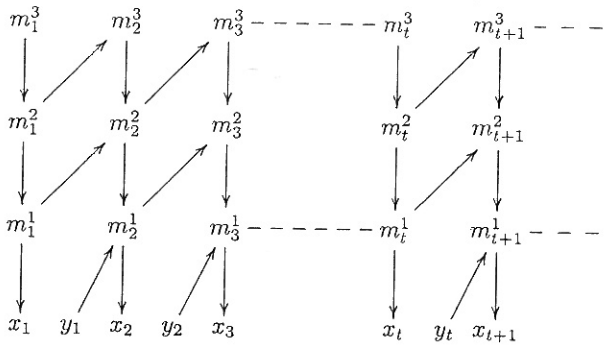
In the HHMM model considered, each memory state receives an information from the current upper-level state and the previous lower-lovel state. This model guarantees a propagation of the information between the several level of the hierarchy, but this propagation is slower than in the case of figure 7. More precisely, this model being denoted h, the probability $h(x|y)$ conditional to the observation takes the form:

$$h(x|y) = \sum_{m_{1:T}^{1:\Lambda} \in M^{1:\Lambda}} \prod_{t=1}^{T} h^0(x_t|m_t^1) h^1(m_t^1|y_{t-1}, m_t^2) \times \prod_{\lambda=2}^{\Lambda} h^\lambda(m_t^\lambda|m_{t-1}^{\lambda-1}, m_t^{\lambda+1}) ,$$

where m^λ is the variable, or memory, for the hidden level λ of the HHMM, M^λ is the set of possible states for the variable m^λ, and Λ is the number of levels of the HHMM. The law h is described graphically in figure 8. It is noteworthy that this model is equivalent to a simple HMM when $\Lambda = 2$. And when $\Lambda = 1$, the law h just maps the immediate observation to action, without any memory of the past observations.

Defining $P[h]$ the complete probabilitic law of the system

Fig. 8. HHMM model for the planning



Universe/Planner:

$$P[h](x,y,z,m) = P(y_{1:T},z_{1:T}|x_{1:T}) \prod_{t=1}^{T} h^0(x_t|m_t^1)$$
$$\times h^1(m_t^1|y_{t-1},m_t^2) \prod_{\lambda=2}^{\Lambda} h^\lambda(m_t^\lambda|m_{t-1}^{\lambda-1},m_t^{\lambda+1}) ,$$

our *approximated* planning problem reduces to find the sub-optimal strategy h_O such that:

$$h_O \in \arg\max_h \sum_{x_{1:T}} \sum_{y_{1:T}} \sum_{z_{1:T}} \sum_{m_{1:T}^{1:\Lambda}} P[h](x,y,z,m)$$
$$\times V(x_{1:T},y_{1:T},z_{1:T}) .$$

A solution to this problem, by means of the cross-entropy method, is proposed in the next section.

IV. Cross-entropic optimization of h

Optimizing h_O means tuning the parameter h in order to tighten the probability $P[h]$ around optimal values for V. This is a *rare event* search problem. The *Cross-Entropy* methods are really adapted for such problems[1]. CE methods are *selection algorithms* closely related to the EM algorithm. It does not need much hypothesis. However, as for many selection algorithm, it is assumed that the evaluation function V is easily computable. Typically, the definition of V may be recursive, *eg.*:

$$V(x,y,z) = \{v_t(x_t,y_t,z_t,\}_{t=T}^{2} v_1(x_1,y_1,z_1)\ \{\})_{t=T}^{2} .$$

Let the *selective rate* ρ be a positive number such that $\rho < 1$. The cross-entropy optimization method follows the synopsis:

1) Initialize h. For example a flat h,
2) Make N tossing $\theta^n = (x^n, y^n, z^n, m^n)$ according to the law $P[h]$,
3) Choose the ρN best samples θ^n according to the evaluation $V(x_{1:T},y_{1:T},z_{1:T})$. Denote S the set of the selected samples,
4) Update h as the minimizer of the cross-entropy with the selected examples:

$$h \in \arg\max \sum_{n\in S} \ln P[h](\theta^n) , \qquad (2)$$

5) Reiterate from step 2 until convergence.

The maximization (2) is easy, in general. Typically, without any further constraint on h and for $2 \le \lambda \le \Lambda$, the maximization (2) is solved by:

$$h^\lambda(A|B,C) = \frac{\mathrm{card}\left\{n \in S, t \,/\, A = m_t^{\lambda,n}, \right.}{\mathrm{card}\left\{n \in S, t \,/\, B = m_t^{\lambda-1,n} \right.} \quad \left. \begin{array}{r} B = m_t^{\lambda-1,n} \text{ and } C = m_{t-1}^{\lambda+1,n} \\ \text{and } C = m_{t-1}^{\lambda+1,n} \end{array} \right\} .$$

The next section presents an example of implementation of this algorithm.

V. Implementation

The algorithm has been applied to a simplified target detection problem. At this time, the *open-loop* implementation has not been investigated yet.

A. Problem setting

A target R is moving in a lattice of 20×20 cells. R is tracked by two mobiles, B_1 and B_2, controled by the planner. B_1 and B_2 have a very limited information about the target position, and are maneuvering much slower:

- A move for B_i is either: *turn left*, *turn right*, *go forward*, *no move*. These moves cannot be combined in a single turn. No diagonal forward,
- The mobiles are initially positioned in the down corners,
- B_i observes whether the target relative position is forward or not,
- B_i knows whether its distance with the target is less than 3 or not.

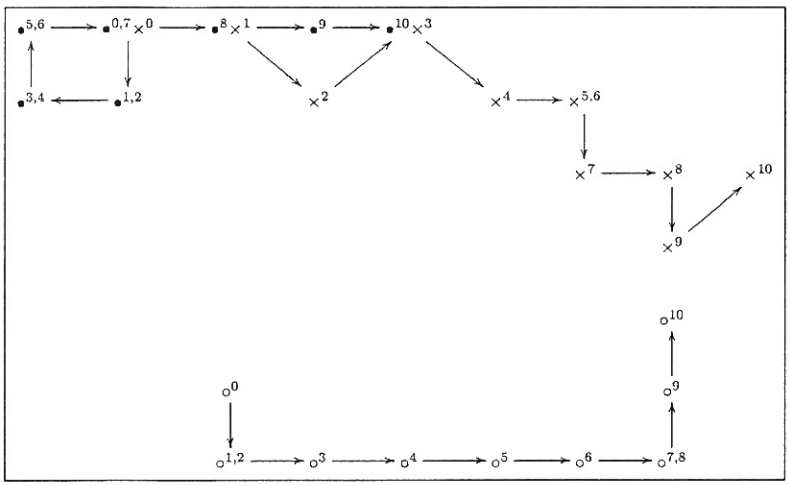
R chooses stochastically its next position in its neighborhood. Any move is possible (up/down, left/right, diagonals, no move). Escape moves are favorized: the probability to choose a position is proportional to its squared distance from the mobiles.

The objective of the planner is to maintain the target sufficiently closed to at least one mobile (in this example, the distance between the target and a mobile is required to be less than 3). More precisely, the evaluation function, V, is just counting the number of such "encounter". The total number of turn is $T = 100$.

B. Results

d) Generality: Like many stochastic algorithm, this algorithm needs some time for convergence. For the considered example, about two hours were needed for convergence (on a 2GHz PC). This speed depends on the size of the HHMM model and on the convergence criterion. In our example, a weak and a strong criterion are used. Within the weak criterion, the algorithm is halted after 100 successive unsuccessful try. Within the strong criterion, the algorithm is halted after 500 successive unsuccessful try. Of course, the strong criterion computes a (slightly) better optimum that the weak criterion. For the same HHMM model, the computed optimal values

Fig. 9. Optimal control sequence



× = target
• = observer 1
○ = observer 2

Relative times are put in supscript

do not depend on the algorithmic instance (small variations result however from the stochastic nature of the algorithm): the convergence seems to be global.

e) Case 1. R does not move: This example has been considered in order to test the algorithm. The position of the target is fixed in the center of the square space. The optimal strategy is known and its value is 87: the time needed to reach the target is 13, and no further move is needed. The learned h_O approximates the evaluation 86. The convergence is rather good.

f) Case 2. R is moving but the observation y is hidden: Initially, R is located within the 20×10 upper cells of the lattice, accordingly to a uniform probabilistic law. The computed optimal means evaluation is about 32. In this case, the mobiles tend to move towards the upper corners.

g) Case 3. R is moving and y is observed: Again, R is located uniformly within the 20×10 upper cells of the lattice. The computed optimal means evaluation is about 69. This evaluation has been obtained from a large HHMM model ($\Lambda = 2$ with 256 states per level, *ie.* $\mathrm{card}(M^\lambda) = 256$) and with the strong criterion. The mobiles strategy results in a tracking of the target. The figure 9 illustrates a short sequence of escape/tracking of the target.

Specific computations are now presented, depending on the number of levels Λ and the number of states per levels. For each case, the weak criterion has been used.

Subcase $\Lambda = 1$. For such model, the action x_t is constructed only from the immediate last observation y_{t-1}. The model does not keep any memory of the past observations. Then, only 16 states are sufficient to describe the hidden variable m_t^1, *ie.* $\mathrm{card}(M^1) = 16$. The resulting optimum is 54.

Subcase $\Lambda = 2$. This model is equivalent to a HMM. The following table gives the computed optimum for several choice of the set of states:

$\mathrm{card}(M^\lambda)$	16	32	64	256
Evaluation	65	66	67	67

It is noteworthy that the memory of the past observations allows better strategies than just immediate observations (case $\Lambda = 1$). Indeed, the evaluation jumps from 54 up to 67.

Subcases $\Lambda > 2$ *have not been sufficiently investigated.* However, although the algorithm still converges, it seems that higher hierarchies do not make a great enhancement in comparison with just a HMM for such a simple example.

VI. Conclusion.

In this paper, we proposed a method for approximating the optimal planning in a partially observable control problem. At this time, the method has been applied to a strictly discrete-state problem and has been seen to work properly. The convergence seems to be global and the optimal HHMMs are able to "track" the target.

The cross-entropic principle could be applied for optimizing continuous laws. It is thus certainly possible to consider the planning of mixed continuous/discrete problems, which are more realistic. At last, many refinement are foreseeable about the structure of the HHMM models. This work is just preliminary and future works should investigate these questions.

References

[1] De Boer and Kroesse and Mannor and Rubinstein, *A Tutorial on the Cross-Entropy Method*, `http://www.cs.utwente.nl/ ptdeboer/ce/`

[2] Richard Bellman, *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.

[3] Edward J. Sondik, *The Optimal Control of Partially Observable Markov Processes*, PhD thesis, Stanford University, Stanford, California, 1971.

[4] Anthony Rocco Cassandra, *Exact and approximate algorithms for partially observable Markov decision processes*, PhD thesis, Brown University, Rhode Island, Providence, May 1998.

[5] Shai Fine and Yoram Singer and Naftali Tishby, *The Hierarchical Hidden Markov Model: Analysis and Application*, Machine Learning, 1998.

[6] Kevin Murphy and Mark Paskin, *Linear Time Inference in Hierarchical HMMs*, Proceedings of Neural Information Processing Systems, 2001.