J Glob Optim (2007) 37:541–555
DOI 10.1007/s10898-006-9061-9

ORIGINAL PAPER

Cross-entropic learning of a machine for the decision in a partially observable universe

Frédéric Dambreville

Received: 19 June 2006 / Accepted: 22 June 2006 / Published online: 9 August 2006

Abstract In this paper, we are interested in optimal decisions in a partially observable universe. Our approach is to directly approximate an optimal strategic tree depending on the observation. This approximation is made by means of a parameterized probabilistic law. A particular family of Hidden Markov Models (HMM), with input *and* output, is considered as a model of policy. A method for optimizing the parameters of these HMMs is proposed and applied. This optimization is based on the cross-entropic (CE) principle for rare events simulation developed by Rubinstein.

Keywords Control · Markov Decision Process/Partially Observable Markov Decision Process · Hierarchical Hidden Markov Models · Bayesian networks · Cross-entropy

Notations Some specific notations are used in this document.

- The variables d, y, x, and m are used for the decision, observation, world state, and machine memory.
- The time t is starting from stage 1 to the maximal stage T. Variables with subscript outside this scope are synonymous to $\emptyset$. For example, $\prod_{t=1}^{T} \pi(x_t|x_{t-1})$ means $\pi(x_1|\emptyset) \prod_{t=2}^{T} \pi(x_t|x_{t-1})$, i.e. a Markov chain. A similar principle is used for the *level* superscript λ in the definition of hierarchical Hidden Markov Models (HHMM).
- The generic notation for a probability is P. However, the functions π and h denote some specific components of the probability. π is a stochastic policy, i.e. a law of the decision conditionally to the observation. h is an approximation of π by a Hidden Markov Models (HMM) family. The hidden state of h is defined as the machine memory m.

F. Dambreville (✉)
Délégation Générale pour l'Armement, DGA/DET/CEP/ASC/GIP
16 Bis avenue Prieur de la Côte d'Or, F 94114, France
e-mail: jogo@fredericdambreville.com

1 Introduction

There are different degrees of difficulty in planning and control 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 decision 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 states are only partially known or the resulting 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 near-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 on the problem to be 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 [2] could be efficiently applied; c.f. Markov Decision Process theory (MDP). This solution has been extended to the case of partial observation, but this solution is generally not practicable, owing to the huge dimension of the variables [10, 4]; c.f. Partially Observable Markov Decision Process (POMDP).

For such reason, different methods for approximating this problem has been introduced. For example, reinforcement learning methods [11] are able to learn an evaluation table of the decision conditionally to the known universe states and an observation short range. In this case, the range of observation is indeed limited in time, because of an exponential grow of the table to learn. Recent works [1] are investigating the case of hierarchical RL, in order to go beyond this range limitation. Whatever, these methods are generally based on an additivity hypothesis about the reward. Another viewpoint is based on the direct learning of the policy [7]. Our approach is of this kind. It is particularly based on the cross-entropy (CE) optimization algorithm developed by Rubinstein and Kroese [9]. This simulation method relies both on a probabilistic modeling of the policies (in this paper, these models are Bayesian networks) and on an efficient and robust iterative algorithm for optimizing the model parameters. More precisely, the policy will be modeled by conditional probabilistic law, i.e. decisions depending on observations, which are involving memories; typically hidden Markov models (HMM) are used. Also is implemented a hierarchical modeling of the policies by means of hierarchical hidden Markov models (HHMM).

The next section introduces some formalism and gives a quick description of the optimal planning in partially observable universes. It is proposed a near-optimal planning method, based on the direct approximation of the optimal decision tree. The third section introduces the family of HHMM being in use for approximating the 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. The CE method is described and applied. The fifth section gives an example of application. A comparison with a reinforcement learning method, the Q-learning, is made. The paper is then concluded.

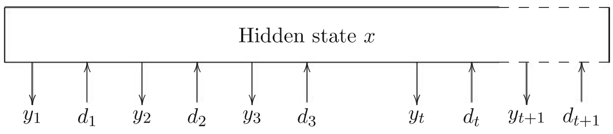

Fig. 1 The world

2 Decision in a partially observable universe

It is assumed that a subject is acting in a given world with a given purpose or mission. Thus, the subject interacts with the world and perceives partial informations. The goal is to optimize the accomplishment of the mission, which is characterized by its reward. The forthcoming paragraphs are formalizing what is actually a world, what is a mission reward, and how is defined an optimal policy for such a mission.

The world. The world is described by an hidden state x, which evolves with the time t; in this paper, the time is discretized and increases from step 1 to step T. More specifically, the variable x_t contains an information which characterizes entirely the world at time t. *In the example of Sect.* 5*, the hidden state is characterized by the locations of the target and patrols.* The evolution of the hidden state is given by the vector $x = x_{1:T} = x_1, \dots, x_t, \dots, x_T$. During the mission, the subject produces decisions $d = d_{1:T}$, which will impact the evolution of the world. *In example* 5*, d is the move of the patrols.* The subject perceives partial observations from the world, denoted $y = y_{1:T}$, which are noisily derived from the hidden state. *In the example, this observation is an inaccurate estimate of the target location.* As a conclusion, the world is characterized by a law describing the hidden states and observations conditionally to the decisions. This *probabilistic* law is denoted P:

> The hidden state x_t and observation y_t are obtained from the law $P(x_t, y_t | x_{1:t-1}, y_{1:t-1}, d_{1:t-1})$, which are conditioned by the past hidden states, observations and decisions. *It is assumed that d_t is generated by the subject after receiving y_t.*

In this paper, the law P is quite general, and for example there is no Markovian hypothesis (this hypothesis is required for a dynamic programming approach). Nevertheless, it is assumed that $P(x_t, y_t | x_{1:t-1}, d_{1:t-1})$ may be sampled very quickly. The law $P(x, y|d)$ is shown in by Fig. 1 . In this figure, the out-going arrows are related to the data produced by the world, i.e. observations, while incoming arrows are for the data consumed by the world, i.e. the decisions. The variables are put in chronological order from left to right: y_t happens before d_t since decision d_t is produced after observing y_t . From now on, $P(x, y|d)$ denotes the law of the world for the completed mission:

$$P(x, y|d) = \prod_{t=1}^{T} P(x_t, y_t | x_{1:t-1}, y_{1:t-1}, d_{1:t-1}).$$

Reward and optimal planning. The mission is limited in time and is characterized by a reward. This reward, denoted $V(d, y, x)$, is a function of the trajectories d, y, x . Typically, the function V could be used for computing the time needed for the mission accomplishment. *The only hypothesis about V is that it is quickly computable.* In particular, the additivity of the reward[1] with time, a requested hypothesis for many classical methods, is not necessary.

[1] Additive reward are of the form $V(d, y, x) = \sum_{t=1}^{T} V_t(d_t, y_t, x_t)$

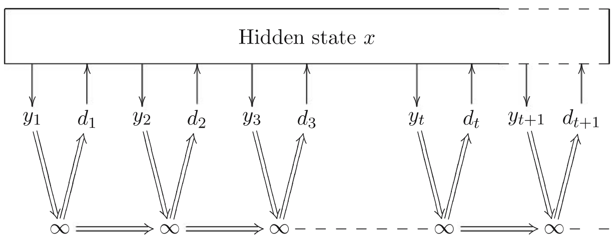

Fig. 2 The optimization process

The purpose is to construct an optimal decision tree $y \mapsto \left(d_t(y_{1:t})|_{t=1}^T\right)$, depending on the past observations, in order to maximize the *mean reward*:

$$\text{Find } d_o \in \arg\max_d \sum_y \sum_x P\left(x,y \left| \left(d_t(y_{1:t})|_{t=1}^T\right)\right.\right) V\left(\left(d_t(y_{1:t})|_{t=1}^T\right),y,x\right). \tag{1}$$

This optimization process is shown in by Fig. 2. The double arrows are related to the variables to be optimized. These arrows describe the information flow between observations and decisions. *The cells denoted ∞ are making decisions and transmitting all the received and generated informations*. This architecture illustrates that planning with observation is a non-finite memory problem : the decision depends on the whole past observations. Since the optimum for such a problem is generally intractable, it is necessary to search for near-optimal solutions. The alternative method proposed now relies on the optimal tuning of a probabilistic model of the policies.

Approximating the decision tree. In a program like (1) , the optimum d_o is a *deterministic* object. In this precise case, d_o is a tree of decision, that is a function which maps to a decision d_t from any sequence of observation $y_{1:t-1}$. But it is more interesting to have a probabilistic viewpoint, when approximating. Then the problem is equivalent to finding $\pi(d|y)$, a probabilistic law of the decisions conditionally to the *past* observations, which maximizes the mean reward:

$$V(\pi) = \sum_d \sum_y \sum_x \prod_{t=1}^T \pi(d_t|d_{1:t-1},y_{1:t})P(x,y|d)\ V(d,y,x).$$

This new problem is still shown in Fig. 2, but the double arrows are now describing a Bayesian network structure for the law π. By the way, there is not a great difference with the deterministic case for the optimum: when d_o is unique, the optimal law $\pi_o \in \arg\max_\pi V(\pi)$ is a Dirac on d_o . However, the probabilistic viewpoint is more suitable to an approximation: it is simpler to handle probabilistic models than deterministic decision trees, and the optimization is ensured to be continuous; moreover, a natural approximation of π_o is obtained by replacing the non-finite memories ∞ by finite memories m (c.f. Fig. 3). Restricting the memory size of the policies is equivalent to approximate the law π by a HMM. Then, the approach developed in this paper is quite general and can be split up into two processes:

- Define a family of parameterized HMMs $\mathcal{H}$.
- Optimize the parameters of the HMM in order to maximize the mean reward:

$$\text{Find} \quad h_o \in \arg\max_{h\in\mathcal{H}} V(h).$$

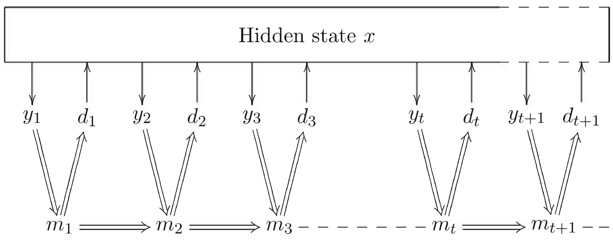
Hidden state x
y_1 d_1 y_2 d_2 y_3 d_3 y_t d_t y_{t+1} d_{t+1}
m_1 m_2 m_3 m_t m_{t+1}

Fig. 3 Finite-memory approximation

As will be seen later, it is easy to tune a HMM optimally by the CE method of Rubinstein and Kroese [9]. But first, it is discussed in the next section about the choice of the family $\mathcal{H}$.

3 Models

General points. The choice of the family of policy models, $\mathcal{H}$, will profoundly impact the efficiency of the approximation. In particular, the models will be characterized by the memory size and the internal structure of the HMMs (e.g. is it hierarchical or not?). Both characteristics will act upon the convergence, as will be seen in the experiments. In the most simple case, the HMMs of $\mathcal{H}$ contain no structure and are distinguished by their memory size only. Example of simple HMM:

> Let M be indeed a finite set of states, describing the memory capacity of our models. Then, the memory of the HMM at time t is $m_t \in M$, a variable valued within M. A HMM $h \in \mathcal{H}$ is thus typically defined by:
>
> $$h(d|y) = \sum_{m \in M^T} h(d,m|y),$$
> $$h(d,m|y) = \prod_{t=1}^{T} \big(h_d(d_t|m_t)h_m(m_t|y_t,m_{t-1})\big),$$
>
> where the conditional law h_d and h_m are time invariant.
> But subsequently will be considered the impact of both the memory and HMM structures. For this purpose a specific family of hierarchical HHM will be introduced and studied. HHMM are indeed a particular case of HMM, implementing strong intern structures.

Hierarchical HMM. Hierarchical models are inspired from biology: to solve a complex problem, factorize it and make decisions in a hierarchical fashion. Low-hierarchies manipulate low-level informations and actions, making short-term decisions. High-hierarchies manipulate high-level informations and actions (uncertainty is less), making long-term decisions. HHMM are such kind of models. *A HHMM is a HMM, which output is either a HMM or an actual output.* A HHMM could also be considered as a hierarchy of *stochastic* processes calling sub-processes. From this common definition, HHMM are complex structures, which are difficult to formalize and to computerize. Nevertheless, these models have been introduced and applied for handwriting recognition [5], as well for modeling complex worlds in control applications

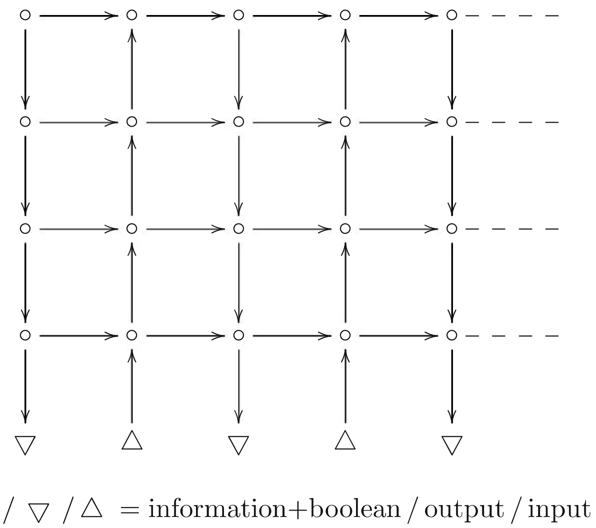

Fig. 4 Model of a controlled HHMM

[12]. A fundamental contribution has been made by Murphy and Paskin [8], which have shown how HHMM could be interpreted as a particular 2-dimension dynamic Bayesian network (DBN). Now, DBN are easily formalized, manipulated, and computerized. The DBN could be considered as HMM with complex intern structures. From the work of Murphy and Paskin, it could be shown that a hierarchical HMM (with input and output) could be interpreted by a DBN as described in Fig. 4, with discrete or semi-continuous states. It appears, that there is a up and down flow of the information between the hierarchical levels in addition to the usual temporal flow (the Markovian property). *It is important to note that Boolean informations are necessary for implementing the hierarchy.* These Boolean are needed for controlling the information flows between processes and subprocesses. The next paragraph introduces the customized model of HHMM, which has been considered in this work. It is simplification of the general HHMM model, and it allows a more simple implementation.

Implemented model. The implemented model family $\mathcal{H}$ is composed by HHMM with Λ hierarchical levels. Each level $\lambda \in [\![1,\Lambda]\!]$ is associated to a finite memory set M^λ (the memory size may change with the hierarchy). The exchange of information between the levels is characterized by the DBN shown in Fig. 5. Notice that each memory cell receives an information from the current upper-level cell and the previous lower-level cell. As a consequence, the hierarchical and temporal information exchanges are guaranteed. In a more formal way, the HHMM $h \in \mathcal{H}$ are of the form:

$$h(d|y) = \sum_{m \in M^{\Lambda T}} h(d,m|y),$$

$$h(d,m|y) = \prod_{t=1}^{T} h^0(d_t|m_t^1) h^1(m_t^1|y_t,m_t^2) \prod_{\lambda=2}^{\Lambda} h^\lambda(m_t^\lambda|m_{t-1}^{\lambda-1},m_t^{\lambda+1}),$$

where $m^\lambda \in M^\lambda$ is the variable for the memory at level λ. It is noteworthy that this model is equivalent to a simple HMM when $\Lambda = 2$. And when $\Lambda = 1$, the law h

$$\begin{tikzcd}
& m_1^3 \arrow[d] & & m_2^3 \arrow[d] & & m_3^3 \arrow[d] \arrow[rrr, dashed, no head] & & & m_t^3 \arrow[d] & & m_{t+1}^3 \arrow[d] \arrow[r, dashed, no head] & {} \\
& m_1^2 \arrow[d] \arrow[urr] & & m_2^2 \arrow[d] \arrow[urr] & & m_3^2 \arrow[d] & & & m_t^2 \arrow[d] \arrow[urr] & & m_{t+1}^2 \arrow[d] & \\
& m_1^1 \arrow[d] \arrow[urr] & & m_2^1 \arrow[d] \arrow[urr] & & m_3^1 \arrow[d] \arrow[rrr, dashed, no head] & & & m_t^1 \arrow[d] \arrow[urr] & & m_{t+1}^1 \arrow[d] \arrow[r, dashed, no head] & {} \\
y_1 \arrow[ur] & d_1 & y_2 \arrow[ur] & d_2 & y_3 \arrow[ur] & d_3 & & y_t \arrow[ur] & d_t & y_{t+1} \arrow[ur] & d_{t+1} &
\end{tikzcd}$$

Fig. 5 HHMM model for the planning

just maps the immediate observation to decisions, without any memory of the past observations.

For any $h \in \mathcal{H}$, define $P[h]$ the complete probabilistic law of the system world/subject:

$$P[h](d,y,x,m) = P(y,x|d)h(d,m|y)$$

Then the issue is to find the near-optimal strategy $h_o \in \mathcal{H}$ such that:

$$h_o \in \arg\max_{h\in\mathcal{H}} \sum_{d,y,x,m} P[h](d,y,x,m)V(d,y,x).$$

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

4 Cross-entropic optimization of *h*

The reader interested in CE methods should refer to the tutorial [3] and the book [9] on the CE method. The CE algorithms were first dedicated to estimating the probability of rare events. A slight change of the basic algorithm made it also good for optimization. In their new article, Homem-de-Mello and Rubinstein [6] have given some results about the global convergence. In order to ensure such convergence, some refinements are introduced particularly about the selective rate.

This presentation is restricted to the basic CE method. The new improvements of the CE algorithm proposed in [6] have not been implemented, but the algorithm has been seen to work properly. For this reason, this paper does not deal with the choice of the selective rate.

4.1 General cross entropy algorithm for the optimization

The CE algorithm repeats until convergence the three successive phases as follows:

1. Generate samples of random data according to a parameterized random mechanism.
2. Select the best samples according to a reward criterion.
3. Update the parameters of the random mechanism, on the basis of the selected samples.

In the particular case of CE, the update in phase 3 is obtained by minimizing the Kullback–Leibler distance, or CE, between the updated random mechanism and the selected samples. The next paragraphs describe on a theoretical example how such method can be used in an optimization problem.

Formalism. Let be given a function $x \mapsto f(x)$; this function is easily computable. The value $f(x)$ has to be maximized, by optimizing the choice of $x \in X$. The function f will be the reward criterion.

Now let be given a family of probabilistic laws, $P_\sigma|_{\sigma\in\Sigma}$, applying on the variable x. The family P is the parameterized random mechanism. The variable x is the random data.

Let $\rho \in]0,1[$ be a selective rate. The CE algorithm for (x, f, P) follows the synopsis:

1. Initialize $\sigma \in \Sigma$.
2. Generate N samples x_n according to P_σ.
3. Select the ρN best samples according to the reward criterion f.
4. Update σ as a minimizer of the CE with the selected samples, i.e.:

$$\text{Find } \sigma \in \Sigma \text{ which maximizes } \sum_{n \text{ selected}} \ln P_\sigma(x_n).$$

5. Repeat from step 2 until convergence.

This algorithm requires f to be easily computable and the sampling of P_σ to be fast.

Interpretation. The CE algorithm tightens the law P_σ around the maximizer of f. Then, when the probabilistic family P is well suited to the maximization of f, it becomes equivalent to find a maximizer for f or to optimize the parameter σ by means of the CE algorithm. The problem is to find a good family ... Another issue is the criterion for deciding the convergence. Some answers are given in [6]. Now, it is outside the scope of this paper to investigate these questions precisely. Our criterion was to stop after a given threshold of successive *unsuccessful tries* and this very simple method has worked fine on our problem.

4.2 Application

Optimizing $h \in \mathcal{H}$ means tuning the parameter h in order to tighten the probability $P[h]$ around the optimal values for V. This is exactly solved by the CE optimization method. However, it is required that the reward function V is easily computable. Typically, the definition of V may be recursive, e.g.:

$$V(d,y,x) = V_T, \quad V_t = v_t(d_t, y_t, x_t, V_{t-1}) \quad \text{and} \quad V_0 = 0.$$

Let the *selective rate* ρ be a positive number such that $\rho < 1$. The CE method for optimizing h follows the synopsis:

1. Initialize h. For example a flat h.
2. Build N samples $\theta^n = (d^n, y^n, x^n, m^n)$ according to the law $P[h]$.
3. Choose the ρN best samples θ^n according to the reward $V(d^n, y^n, x^n)$. Denote S the set of the selected samples.
4. Update h as the minimizer of the CE with the selected samples, i.e.:

$$\text{Find } h \in \mathcal{H} \text{ which maximizes } \sum_{n\in S} \ln P[h](\theta^n). \tag{2}$$

5. Reiterate from step 2 until convergence.

For our HHMM model, the maximization (2) is solved by:

$$h^0(A|B) = \frac{\operatorname{card}\left\{n \in S, t \,/\, A = d_t^n \text{ and } B = m_t^{1,n}\right\}}{\operatorname{card}\left\{n \in S, t \,/\, B = m_t^{1,n}\right\}},$$

$$h^1(A|B,C) = \frac{\operatorname{card}\left\{n \in S, t \,/\, A = m_t^{1,n}, B = y_t^n \text{ and } C = m_t^{2,n}\right\}}{\operatorname{card}\left\{n \in S, t \,/\, B = y_t^n \text{ and } C = m_t^{2,n}\right\}}$$

and for $2 \leq \lambda \leq \Lambda$:

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

The next section presents an example of implementation of the algorithm described in Sect. 4.2.

5 Implementation

The algorithm has been applied to a simulated target detection problem.

5.1 Problem setting

A target R is moving in a lattice of 20×20 cells, i.e. $[\![0, 19]\!]^2$. R is tracked by two mobiles, B and C, controlled by the subject. The coordinate of R, B and C at time t are denoted (i_R^t, j_R^t), (i_B^t, j_B^t) and (i_C^t, j_C^t). B and C have a very limited information about the target position, and are maneuvering much slower as follows:

- A move for B (respectively C) is either: *turn left*, *turn right*, *go forward*, *no move*. Consequently, there are $4 \times 4 = 16$ possible actions for the subject. These moves cannot be combined in a single turn. No diagonal forward: a mobile is either directed up, right, down or left.
- The mobiles are initially positioned in the down corners, i.e. $i_B^1 = 0$, $j_B^1 = 19$ and $i_C^1 = 19$, $j_C^1 = 19$. The mobile are initially directed *downward*.
- B (respectively C) observes whether the target relative position is forward or not. More precisely:
  (1) when B is directed upward, it knows whether $j_R < j_B$ or not,
  (2) when B is directed right, it knows whether $i_R > i_B$ or not,
  (3) when B is directed downward, it knows whether $j_R > j_B$ or not,
  (4) when B is directed left, it knows whether $i_R < i_B$ or not,
- B (respectively C) knows whether its distance with the target is less than 3, i.e. $d_\infty(B,R) < 3$, or not. The distance d_∞ is defined by:

$$d_\infty(B,R) = \max\{|i_B - i_R|, |j_B - j_R|\}.$$

At last, there are $2^4 = 16$ possible observations for the subject.

Several test cases have been considered. In case 1, the target R does not move. In any other case, the target R chooses stochastically its next position in its neighborhood. Any move is possible (up/down, left/right, diagonals, no move). The probability to choose a new position is proportional to the sum of the squared distance from the mobiles:

$$\begin{aligned} &P(R^{t+1}|R^t) = 0 && \text{if } |i_R^{t+1} - i_R^t| > 1 \\ & && \text{or } |j_R^{t+1} - j_R^t| > 1, \\ &P(R_{t+1}|R_t) \propto (i_R^{t+1} - i_B^t)^2 + (j_R^{t+1} - j_B^t)^2 && \\ &\qquad\qquad + (i_R^{t+1} - i_C^t)^2 + (j_R^{t+1} - j_C^t)^2 && \text{else.} \end{aligned}$$

This definition was intended to favor escape moves: more great is a distance, more probable is the move. But in such summation, a short distance will be neglected compared to a long distance. It is implied that a distant mobile will hide a nearby mobile. This “deluding” property will induce actually two different kinds of strategy, within the learned machines.

The objective of the subject 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 not more than 3). More precisely, the reward function, V, is just counting the number of such “encounter:”

$$V_0 = 0, \qquad V_t = V_{t-1} + 1 \quad \text{if } d_\infty(B^t, R^t) \le 3 \quad \text{or } d_\infty(C^t, R^t) \le 3, \quad V_t = V_{t-1} \text{ else.}$$

The total number of turns is $T = 100$.

5.2 Results

Generality. Like many stochastic algorithms, this algorithm needs some time for convergence. For the considered example, about 2 h were needed for convergence (on a 2 GHz PC); the selective rate was $\rho = 0.5$. This speed depends on the size of the HHMM model and on the convergence criterion. A weak and a strong criterion are used for deciding the convergence. Within the weak criterion, the algorithm is terminated after 100 successive unsuccessful tries. Within the strong criterion, the algorithm is terminated after 500 successive unsuccessful tries. Of course, the strong criterion computes a (slightly) better optimum than the weak criterion, but it needs time. Because of the many tested examples, the weak criterion has been the most used in particular for the big models. For the same HHMM model, the computed optimal values do not depend on the algorithmic instance (small variations result however from the stochastic nature of the algorithm).

In the sequel, mean rewards are rounded to the nearest integer, or are expressed as a percentage of the optimum. Thus, the presentation is made clearer. And owing to the small variations of this stochastic algorithm, more precision turns out to be irrelevant.

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, i.e. $i_R^1 = j_R^1 = 10$. It is recalled that the mobiles are initially directed downward. Then, the optimal strategy is known and its value is 85 : the time needed to reach the target is 15 , and no further move is needed. The learned h_o approximates the reward 84 . The convergence is good.

Case 2 (*R is moving but the observation y is hidden*) Initially, R is located within the 20×10 upper cells of the lattice (i.e. $[\![0, 19]\!] \times [\![0, 9]\!]$), accordingly to a uniform probabilistic law. The computed optimal means reward is about 32. In this case, the mobiles tend to move towards the upper corners.

Case 3 (*R is moving and y is observed*) Again, R^1 is located uniformly within the 20×10 upper cells of the lattice. The computed optimal means reward is about 69. This reward has been obtained from a large HHMM model ($\Lambda = 2$ with 256 states per level, i.e. $\text{card}(M^\lambda) = 256$) and with the strong criterion. However, somewhat smaller models should work as well.

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. The rewards are now expressed as percentage of the computed optimum.

Subcase $\Lambda = 1$. For such model, the action d_t is constructed only from the immediate last observation y_t. 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, i.e. $\text{card}(M^1) = 16$. The resulting reward is 78% of the optimum.

Subcase $\Lambda = 2$. This model is equivalent to a HMM and it is assumed that $\text{card}(M^1) = \text{card}(M^2)$. The following table gives the computed reward for several choices of the memory size:

Card(M^λ)	16	32	64	256
Reward %	94	96	97	97

It is noteworthy that the memory of the past observations allows better strategies than the only last observation (case $\Lambda = 1$). Indeed, the reward jumps from 78 to 97%.

Subcases $\Lambda > 2$. A comparison of graduated hierarchic models, $1 \leq \Lambda \leq 4$, has been made. The first level contained 16 possible states, and the higher levels were restricted to two states:

Hierarchic grade	$\Lambda = 1$	$\Lambda = 2$	$\Lambda = 3$	$\Lambda = 4$
$\text{Card}(M^\lambda)\rvert_{\lambda=1}^{\Lambda}$	16	16, 2	16, 2, 2	16, 2, 2, 2

The test has been accomplished according to the weak criterion:

Hierarchic grade	$\Lambda = 1$	$\Lambda = 2$	$\Lambda = 3$	$\Lambda = 4$
Reward (weak %)	78	85	81	94

and the strong criterion:

Hierarchic grade	$\Lambda = 1$	$\Lambda = 2$	$\Lambda = 3$	$\Lambda = 4$
Reward (strong %)	80	88	93	96

It seems that a high-hierarchic grade (i.e. more structure) makes the convergence difficult. This is particularly the case here for the grade $\Lambda = 3$, which failed under the weak criterion at only 81%. However, the algorithm works again when improving the convergence criterion.

It is interesting to make a comparison with the subcase $\Lambda = 2$ where $\text{card}(M^1) = \text{card}(M^2) = 16$. Under the weak criterion, the result for this HHMM was 94% as for the grade $\Lambda = 4$. However, the dimension of the law is quite different for the two models:

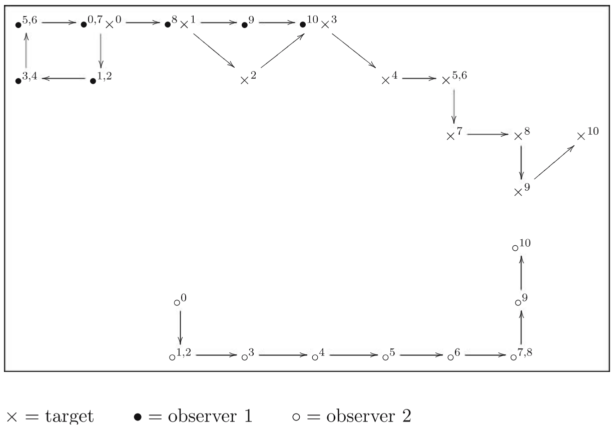

Fig. 6 Near-optimal control sequence

- $15 \times 16 + 15 \times 16 \times 16 + 15 \times 16 = 4{,}320$ for the 2-level HHMM,
- $15 \times 16 + 15 \times 16 \times 2 + 1 \times 16 \times 2 + 1 \times 2 \times 2 + 1 \times 2 = 758$ for the 4-level HHMM.

This dimension is a rough characterization of the complexity of the model. It seems clear on these examples that the highly hierarchized models are more efficient than the weakly hierarchized models. And the problem considered here is quite simple. On complex problems, hierarchical models may be pre-eminent.

Global behavior of the algorithm. The convergence speed is low at the beginning. After this initial stage, it improves greatly until it reaches a new "waiting" stage. This alternation of low speed and great speed stages have been noticed several times.

The near optimal policy. It is now discussed about the behavior of the best found policy. This policy has reach the mean reward 69. The mobiles strategy results in a tracking of the target. The Fig. 6 shows a short sequence of escape/tracking of the target. It has been noticed two quite distinct behaviors, among the many runs of the policy:

- *The two mobiles may both cooperate on tracking the target.*
- *When the target is near a border, one mobile may stay along the opposite border while the other mobile may perform the tracking.* This strategy seems strange at first sight. But it is recalled that the moving rule of the target tends to neglect a nearby mobile compared to a distant mobile. In this strategy, the first mobile is just annihilating the ability of the target to escape from the tracking of the second mobile.

5.3 Comparison with the Q-learning

The Q-learning is a reinforcement learning method, which is based on the computation of a table evaluating the decision conditionally to the *known information*. The known information is typically the state of the world if it is known, or partial states and observations. Since the known information increases exponentially with the observation range, the test will only implement a Q-learning based on the immediate past observation. Now, let us recall some theoretical grounds about the Q-learning.

Theory. A founding reference about reinforcement learning is the well known book of Sutton and Barto [11], which is available on the Internet. This paragraph will not enter deeply into the subject, and is limited to a simple description of the Q-learning. Moreover, we will make the hypothesis of infinite horizon (that is $T = \infty$) with a weak discounting of the reward $\gamma = 0.99$, so as to implement the algorithm in its most classical form. Tests however have also been made with a finite horizon but have not achieved a good convergence for the considered algorithm.

The learning relies on the following hypotheses:

- At each step t, the subject has a (partial) knowledge s of the state of the world, and chooses an action a.
- Let V_{t+1} be the cumulated reward from step $t+1$ to step ∞. Assume a state s_t and action a_t at step t. Then $V_t = R(s_t, a_t) + \gamma V_{t+1}$, i.e. an instantaneous reward R is obtained and cumulated to the discounted future reward.

The question is: *being given a current state s, what is the best action a to be done?* The answer is simple, if we are able to predict the future and evaluate the expected cumulated reward $Q(s, a)$ for any a: the best action is $a_o \in \arg\max_a Q(s, a)$. The following algorithm could be used for learning the table Q (taken from [11]):

```
• Initialize $Q(s, a)$ arbitrary
• (Repeat for each episode: [*finite-horizon case*])
    (1) Initialize $s$
    (2) Repeat for each step (of the episode):
        (a) With probability $1 - \epsilon$ choose $a \in \arg\max_a Q(s, a)$ ; otherwise chose $a$ randomly
        (b) Take action $a$, receive reward $R(s, a)$ and observe the new state $s'$
        (c) Set $Q(s, a) := Q(s, a) + \alpha\big(R(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a)\big)$
        (d) Set $s := s'$
    (3) (until $s$ is terminal)
```

where α controls the convergence speed and ϵ the innovation.
In our implementation, $s = (y_t, i^t_B, j^t_B, i^t_C, j^t_C, \text{directions})$, $a = d_t$, $\alpha = 0.1$, $\epsilon = 1/\ln t$ and the instantaneous reward R is compliant with the experiment definition of previous section. Since s contains the last observation plus the known part of the world state, this experiment should be equivalent to [case 3/subcase $\Lambda = 1$] considered previously. The computer memory needed to store the table Q was approximately 2 giga-byte: we are around the limits of the computer. In particular, it is rather uneasy to involve a greater observation range without some approximations.

Results. The algorithm has been stopped after 10^{11} iterations, but 10^{10} seemed sufficient. It took several hours, but the algorithm has not been optimized. In order to make the comparison possible with our method, the Q-strategies have been evaluated by a non-discounted cumulation of the reward on 100-step-wide windows. Moreover, these evaluations have been made:

- from the initial stage of the simulation, so as to conform to previous section,
- after many cycles, so as to simulate an infinite horizon.

The following table makes a comparison between the Q-strategies and the model based strategies with $\Lambda = 1$.

	Worse %	Mean %	Best %
Q-policy/stage 0	0	40	112
Q-policy/∞-horizon	0	51	145
Model based $\Lambda = 1$	44	78	105

It is first noticed that the policy obtained by the Q-learning is less regulated than the model based policy. Moreover, although it may be quite good to track a target when the encounter has been initialized (best is 145%), it is rather bad at initializing the encounter (mean for initial stage is 40%) or when the tracking is lost (worst is 0%). At last, the mean evaluation at infinite horizon is 51%, which is even smaller than the model-based policy working from the initial stage.

On this example, and for this simple Q-learning implementation, the comparison is favorable to the model-based policy. Moreover, model-based policies are able to manage more observation range. Now, this planning example has been constructed so as to make difficult the management of the state variables (the dimension is huge) and observations (the observations are poor and have to be combined). For such a problem, a more dedicated RL-method should be chosen.

6 Conclusion

In this paper, we proposed a general method for approximating the optimal planning in a partially observable world. The HHMM families have been used for approximating the optimal decision tree, and the approximation has been optimized by means of the CE method.

At this time, the method has been applied to a strictly discrete-state problem and has been seen to work properly. This algorithm has been compared favorably with a Q-learning implementation of the considered problem: it is able to manage more observation range, and the optimized policy is more regulated. An interesting point is that the optimized policy has discovered two quite different global strategies and is able to choose between them: make the mobiles both cooperate on tracking or require one mobile for deluding the target.

The results are promising. However, the observation and action spaces are limited to a few number of states. And what happens if the hidden space becomes much more intricate? There are several possible answers to such difficulties:

First, the CE principle could be applied for optimizing continuous laws. It is thus certainly possible to consider semi-continuous models, which will be more realistic for a planning policy. Second, many refinements are foreseeable about the structure of the models. Hierarchic models for observation, decision, and memory should be improved in order to locally factorize intricate problems. This research is just preliminary and future works should investigate these questions.

References

1. Bakker, B., Schmidhuber J.: Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In Proceedings of the 8th Conference on Intelligent Autonomous Systems, pp. 438–445. Amsterdam, The Netherlands (2004)
2. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton, New Jersey (1957)

3. de Boer, P.-T., Kroesse, D.P., Mannor, S., Rubinstein, R.Y.: A tutorial on the cross-entropy method, http://www.cs.utwente.nl/~ptdeboer/ce/
4. Cassandra, A.R.: Exact and approximate algorithms for partially observable Markov decision processes. PhD thesis, Brown University, Rhode Island, Providence (1998)
5. Fine, S., Singer, Y., Tishby, N.: The hierarchical hidden markov model: Analysis and Application. Machine Learning **32**(1), 41–62 (1998)
6. Homem-de-Mello, T., Rubinstein, R.Y.: Rare event estimation for static models via cross-entropy and importance sampling. http://users.iems.nwu.edu/~tito/list.htm
7. Meuleau, N., Peshkin, L., Kim, K-E., Kaelbling, L.P.: Learning finite-state controllers for partially observable environments. In Proc. of UAI-99, pp. 427–436. Stockholm (1999)
8. Murphy, K., Paskin, M.: Linear time inference in hierarchical HMMs. In: Proceedings of Neural Information Processing Systems, Vancouver, Canada (2001)
9. Rubinstein, R., Kroese, D. P.: The Cross-Entropy method. An unified approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning. Information Science & Statistics, Springer Berlin (2004)
10. Sondik, E. J.: The optimal control of partially observable markov processes. PhD thesis, Stanford University, Stanford, California (1971)
11. Sutton, R.J., Barto, A.G.: Reinforcement Learning, MIT Press, Cambridge, MA (2000)
12. Theocharous, G: Hierarchical learning and planning in partially observable markov decision processes. PhD thesis, Michigan State University (2002)