Solving Hidden-Semi-Markov-Mode Markov Decision Problems

Emmanuel Hadoux, Aurélie Beynier, Paul Weng

Solving Hidden-Semi-Markov-Mode Markov Decision Problems

Emmanuel Hadoux, Aurélie Beynier, and Paul Weng

Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, Paris, France
firstname.surname@lip6.fr

Abstract. Hidden-Mode Markov Decision Processes (HM-MDPs) were proposed to represent sequential decision-making problems in non-stationary environments that evolve according to a Markov chain. We introduce in this paper Hidden-Semi-Markov-Mode Markov Decision Processes (HS3MDPs), a generalization of HM-MDPs to the more realistic case of non-stationary environments evolving according to a semi-Markov chain. Like HM-MDPs, HS3MDPs form a subclass of Partially Observable Markov Decision Processes. Therefore, large instances of HS3MDPs (and HM-MDPs) can be solved using an online algorithm, the Partially Observable Monte Carlo Planning (POMCP) algorithm, based on Monte Carlo Tree Search exploiting particle filters for belief state approximation. We propose a first adaptation of POMCP to solve HS3MDPs more efficiently by exploiting their structure. Our empirical results show that the first adapted POMCP reaches higher cumulative rewards than the original algorithm. However, in larger instances, POMCP may run out of particles. To solve this issue, we propose a second adaptation of POMCP, replacing particle filters by exact representations of beliefs. Our empirical results indicate that this new version reaches high cumulative rewards faster than the former adapted POMCP and still remains efficient even for large problems.

1 Introduction

Markov Decision Processes (MDPs) provide a general formal framework for sequential decision-making under uncertainty. They have proved to be powerful for solving many planning problems [14]. However, MDPs run under the assumption that the environment is stationary, i.e., the transition function and/or the reward function do not evolve through time. In many real-world applications, this assumption does not hold and the sources of non-stationarity are diverse. For instance, the environment may change due to external events. In finance, when investing on the stock market, a financial crisis changes the dynamics of stock prices. Another example of non-stationary environment concerns multi-agent systems. Indeed, from the viewpoint of one agent, a change of behavior (e.g., due to learning) of another agent may affect the environment of the first agent.

Planning in a non-stationary environment is a difficult problem to tackle in the general case. We focus instead on a subclass of problems where non-stationary environments evolve according to a small number of non-observable modes, which are modeled as MDPs and represent different possible dynamics and rewards of that environment. An example of problem belonging to this subclass is that of elevator control [6] where the environment can typically be in three modes: morning rush-hour, late-afternoon rush-hour and non-rush-hour. Planning in such non-stationary environments has already been studied in the MDP framework [8] and in the reinforcement learning framework [7, 9, 15]. In all those works, non-stationary environments are represented with multiple modes. The model of Hidden-Mode Markov Decision Processes (HM-MDPs) proposed by Choi et al.[8] formalizes this idea. HM-MDPs constitute a subclass of Partially Observable MDPs. In HM-MDPs, the environmental changes are described by a Markov chain and thus occur at each decision step. However, we argue that this assumption is not always realistic. Indeed, in the elevator problem for instance, allowing, even with a small probability, the environment to be able to change between different rush modes at every move of the elevator is debatable.

In this paper, we propose a natural extension of HM-MDPs, called Hidden-Semi-Markov-Mode Markov Decision Process (HS3MDP), where the non-stationary environment evolves according to a semi-Markov chain. This new model is to hidden semi-Markov models [17] what HM-MDPs are to hidden Makov models. In HS3MDPs, when the environment stochastically changes to a new mode, it stays in that mode during a stochastically drawn duration. While HM-MDPs assume that environmental changes follow a geometric law, this assumption is relaxed in HS3MDPs.

In order to solve large-sized HS3MDPs, we exploit the Partially Observable Monte Carlo Planning (POMCP) algorithm [16], an online algorithm proposed for approximately solving POMDPs, based on Monte Carlo Tree Search and particle filters for belief state approximation. We present two improvements of POMCP for solving HS3MDPs more efficiently. The first adaptation exploits the special structure of HS3MDPs and the second furthermore represents belief states exactly instead of using particle filters. Finally, we experimentally validate those algorithms showing their effectiveness on a diverse range of domains.

In Sect. 2, we recall the necessary notations and definitions. Then, in Sect. 3, we introduce our new model. In Sect. 4, we present two adapted algorithms for solving HS3MDPs. Experimental results are presented in Sect. 5. Finally, we conclude in Sect. 6.

2 Background

Markov Decision Process. A *Markov Decision Process* (MDP) [14] is defined by $\langle \mathbf{S}, \mathbf{A}, T, R \rangle$ where $\mathbf{S}$ is a finite set of states, $\mathbf{A}$ is a finite set of actions, $T(s, a, s')$ is the probability of reaching state s' from s after executing action a and $R(s, a) \in \mathbb{R}$ is the immediate reward obtained after performing action a in s. A *policy* π is a sequence $(\delta_0, \delta_1, \ldots, \delta_t, \ldots)$ of *decision rules* such as each

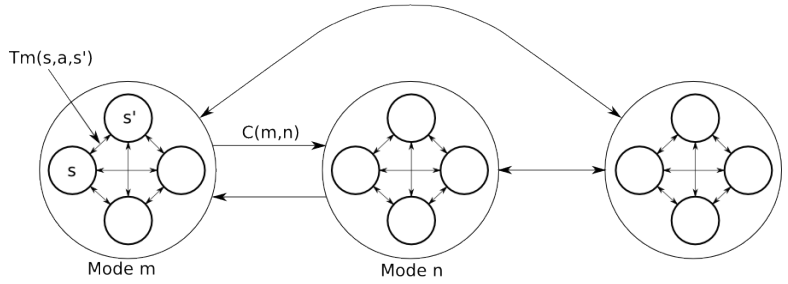

Fig. 1: HM-MDP representation with 3 modes and 4 states

decision rule $\delta_t : \mathbf{S} \to \mathbf{A}$ dictates which action to take for each state at timestep t. In a state s, a policy π can be valued by the expected discounted total reward it yields:

$$V^{\pi}(s) = E_{\pi}(\sum_{t} \gamma^t R(S_t, A_t) \,|\, S_0 = s) \ . \tag{1}$$

where $\gamma \in [0, 1[$ is a discount factor. Function V^{π} is called the *value function* of π. Solving an MDP consists in finding an *optimal policy*, i.e., a policy that maximizes the expected discounted sum of rewards. One of the main limitations of the standard MDP framework is that it requires the transition and reward functions to be stationary.

Hidden-Mode MDP. *Hidden-Mode MDPs* (HM-MDPs) formalize a subclass of non-stationary problems where environmental changes are limited to a fixed and known number n of modes. Each mode represents a possible stationary environment, formalized as an MDP. Transitions between modes represent environmental changes. Formally, an HM-MDP is defined as follows [8]. For $i \in \{1, \ldots, n\}$, let $m_i = \langle \mathbf{S}, \mathbf{A}, T_i, R_i \rangle$ be a mode, i.e., an MDP. An HM-MDP is characterized by $\langle \mathbf{M}, C \rangle$ where $\mathbf{M} = \{m_1, \ldots, m_n\}$ and $C : \mathbf{M} \times \mathbf{M} \to [0, 1]$ is a transition function over modes. Notice that $\mathbf{S}$ and $\mathbf{A}$ are shared by all m_i's and that an HM-MDP with $n = 1$ is a standard MDP. In HM-MDPs, the only observable information is the current state $s \in \mathbf{S}$, the current mode $m \in \mathbf{M}$ is not observable. Figure 1, showing a 3-mode, 4-state HM-MDP, depicts how HM-MDPs can be visualized. To illustrate further the definition of an HM-MDP, we present a simple example:

Example 1. The elevator problem consists in controlling e elevators in a f-floor building. At each decision step, a user may call an elevator at any floor and, once inside, select any desired floor to go. The number of states is then $2^{f(e+1)} \times f^e$. The modes are the different types of rush-hour and therefore have an influence on which buttons are pressed, which is described by different transition functions over states. Three actions can be applied to each elevator: go up/down by one floor and open the doors, leading to an action set of size 3^e. Finally, in this problem, the reward function is identical for all modes, the controller receives a penalty for each unsatisfied user.

Considering an office building of 2 floors with 1 elevator: $\mathbf{M}$ = {morning rush-hour, late-afternoon rush-hour, non-rush-hour}, $\mathbf{S}$ = {1^{st} floor call button states} $\times$ {2^{nd} floor call button states} $\times$ {1^{st} floor drop-off button states} $\times$ {2^{nd} floor drop-off button states} $\times$ {elevator positions}, $\mathbf{A}$ = {open, up, down}. In this small example, there are 32 states, 3 actions and 3 modes. The transition function in the morning rush-hour mode describes the situation where it is more probable for the elevator to be called at the first floor. In the late-afternoon rush-hour mode, it describes the opposite situation where users tend to leave the office. For the non-rush-hour mode, the transition function models the normal operating situation.

Partially Observable MDP. *Partially Observable MDPs* (POMDPs) extends MDPs to partially observable settings [14] and are defined by $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{Q}, \mathcal{R} \rangle$ where $\mathcal{S}$ is a set of POMDP states, $\mathcal{A}$ a set of actions, $\mathcal{O}$ a set of observations, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a transition function over POMDP states, $\mathcal{Q} : \mathcal{S} \times \mathcal{O} \to \mathbb{R}$ is a probability distribution over observations and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function. Since the agent does not observe the POMDP state, she has to act based on her only available information (i.e., at step t, the probability distribution over the initial states and her history of observations and actions up to the current step t) which can be represented as a probability distribution over states, called *belief state* [2]. Optimal algorithms have been proposed to solve POMDPs [10, 3], but they do not scale to large-sized problems. Indeed, finding an optimal policy for infinite-horizon POMDPs is PSPACE-Complete [13].

Choi et al. have shown that an HM-MDP can be seen as a POMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{Q}, \mathcal{R} \rangle$ where $\mathcal{S} = \mathbf{M} \times \mathbf{S}$, $\mathcal{A} = \mathbf{A}$, $\mathcal{O} = \mathbf{S}$, $\mathcal{T}(\langle m, s \rangle, a, \langle m', s' \rangle) = T_m(s, a, s') \times C(m, m')$, $\mathcal{Q}(\langle m, s \rangle, a, o) = 1$ if $s = o$ and 0 otherwise, $\mathcal{R}(\langle m, s \rangle, a) = R_m(s, a)$. They have also proposed algorithms to optimally solve HM-MDPs [6, 8]. They have adapted exact POMDP solving methods in order to exploit the structure of HM-MDPs. Those adapted methods can solve larger instances of HM-MDPs than the original ones, but they also suffer from the curse of dimensionality. Indeed, solving an HM-MDP is still PSPACE-Complete [5]. Like exact POMDP solving algorithms, exact HM-MDP solving algorithms does not scale. In that case, one has to resort to approximate algorithms like POMCP.

POMCP. The *Partially Observable Monte-Carlo Planning* (POMCP) algorithm [16] is one of the most efficient online algorithms to approximately solve large-sized POMDPs. To choose an action at a given timestep, POMCP (Alg. 1) runs an effective version of Monte-Carlo Tree Search (MCTS), called UCT (Upper Confidence Bounds (UCB) applied to Trees) [11], using a black-box simulator of the environment and a particle filter to approximate a belief state. This search tree is built iteratively. POMCP uses the simulator to run a fixed number of simulations in order to evaluate the actions before performing in the real environment the best action found in the search tree. At one decision step, to choose which action to perform, SEARCH(τ) is invoked with τ the current history, i.e., the sequence of past observations and actions. This history can be expanded with an action a giving τa and an observation o giving $\tau a o$. The root of the search tree is a node matching the last seen observation. Its children are

all possible actions, whose own children are the respective possible observations given an action. A node of the tree is a triplet $\langle N(\tau), V(\tau), B(\tau)\rangle$ associated to τ where the components are respectively the number of times τ has been visited, its mean value and the set of particles (i.e., POMDP states) for this history. During a simulation, the algorithm randomly draws a particle p from the particle set $B(\tau)$ and uses the simulator $\mathcal{G}(p, a)$ to get the new particle p', the observation o and the reward r. Actions are selected (Line 19 of Alg. 1) following the UCB1 procedure guaranteeing a good exploration-exploitation compromise. Once all simulations have been done, a step is performed in the real environment with the action returned by SEARCH, i.e., the best action found in the search tree. The algorithm sets the new root to the node matching this observation and prunes the tree.

At the beginning, POMCP is initialized with an empty history and an initial (e.g., uniform) distribution $\mathcal{I}$ over states. Two important parameters have to be set to guarantee that a good action is selected: the tree depth and the number of simulations. The tree depth d can be deduced from the discount factor γ for a given precision $\epsilon > 0$ as follows: $d = \lfloor \log(\epsilon)/\log(\gamma) \rfloor$. The higher the number of simulations, the better the estimation of the values of the actions but the longer it takes to run. This parameter is generally determined by time constraints. However, as the number of simulations tends to infinity, this algorithm is theoretically guaranteed to choose the optimal action at each step. Finally, notice that the size of the initial particle filter is generally set in function of the number of simulations.

3 HS3MDP

The HM-MDP framework is not always the most suitable model for representing sequential decision-making in non-stationary environments as it assumes that the environment may change at every timestep. For instance, modeling the elevator problem with an HM-MDP is problematic as decisions have to be made every (say) second, while a mode (rush hour or not) can last several hours. In a problem where this assumption does not hold, the usual modeling trick is to set a low probability of transition between modes. However, from a theoretical viewpoint, this is more than questionable when mode transitions are not geometrically distributed. One of the main contributions of this paper is to propose a more natural model for such cases where the environment dynamics evolve according to a semi-Markov chain. More precisely, the new model we propose, called *Hidden-Semi-Markov-Mode MDP*, represents environmental changes with hidden semi-Markov models [17] while in HM-MDPs, they were represented with hidden Makov models.

Definition of HS3MDP. Formally, *Hidden-Semi-Markov-Mode Markov Decision Process* (HS3MDP) is defined by $\langle \mathbf{M}, C, H\rangle$ where $\mathbf{M} = \{m_1, \ldots, m_n\}$ is a set of modes, $C : \mathbf{M} \times \mathbf{M} \to [0, 1]$ is a transition function over modes and $H : \mathbf{M} \times \mathbf{M} \times \mathbb{N} \to [0, 1]$ is a *mode duration function*. Transition $C(m, m')$ repre-

Algorithm 1: POMCP

procedure SEARCH(τ)
- **foreach** *simulations* **do**
  - **if** $\tau = empty$ **then**
    - $p \sim \mathcal{I}$
  - **else**
    - $p \sim B(\tau)$
  - SIMULATE$(p, \tau, 0)$
- **return** $\arg\max_b V(\tau b)$

procedure ROLLOUT$(p, \tau, depth)$
- **if** $\gamma^{depth} < \epsilon$ **then**
  - **return** 0
- $a \sim \pi_{rollout}(\tau, \cdot)$
- $(p', o, r) \sim \mathcal{G}(p, a)$
- **return** $r + \gamma$.ROLLOUT$(p', \tau ao, depth + 1)$

procedure SIMULATE$(p, \tau, depth)$
- **if** $\gamma^{depth} < \epsilon$ **then**
  - **return** 0
- **if** $\tau \notin Tree$ **then**
  - **forall the** $a \in \mathbf{A}$ **do**
    - $Tree(\tau a) \leftarrow (N_{init}(\tau a), V_{init}(\tau a), \emptyset)$
  - **return** ROLLOUT$(p, \tau, depth)$
- $a \leftarrow \arg\max_b V(\tau b) + c\sqrt{\frac{\log N(\tau)}{N(\tau b)}}$
- $(p', o, r) \sim \mathcal{G}(p, a)$
- $R \leftarrow r + \gamma$.SIMULATE$(p', \tau ao, depth + 1)$
- $B(\tau) \leftarrow B(\tau) \cup \{p\}$
- $N(\tau) \leftarrow N(\tau) + 1$
- $N(\tau a) \leftarrow N(\tau a) + 1$
- $V(\tau a) \leftarrow V(\tau a) + \frac{R - V(\tau a)}{N(\tau a)}$
- **return** R

sents the probability of moving to new mode m' from current mode m knowing that the *duration* in m (i.e., the number of remaining timesteps to stay in m) is null. Value $H(m, m', h)$ represents the probability of staying h timesteps in new mode m' when the current mode is m. Both the mode and the duration are not observable.

At each timestep, after a state transition in current mode m, the next mode m' and its duration h' are determined as follows:

$$\begin{cases} \text{if } h > 0 \; m' = m \text{ and } h' = h - 1 \\ \text{if } h = 0 \; m' \sim C(m, \cdot) \\ \qquad\qquad h' = k - 1 \text{ where } k \sim H(m, m', \cdot) \end{cases} \tag{2}$$

where h is the duration of current mode m. If h is positive, the environment does not change. But, if h is null, the environment evolves according to transition function C and the number of steps to stay in the new mode is drawn following conditional probability H.

Like HM-MDPs, HS3MDPs form a subclass of POMDPs. An HS3MDP can be reformulated as a POMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{Q}, \mathcal{R} \rangle$ whose components are defined by: $\mathcal{S} = \mathbf{M} \times \mathbf{S} \times \mathbb{N}$, $\mathcal{A} = \mathbf{A}$, $\mathcal{O} = \mathbf{S}$, $\mathcal{T}(\langle m, s, h \rangle, a, \langle m', s', h' \rangle) = \alpha T_m(s, a, s')$ with

$$\alpha = \begin{cases} C(m, m') \times H(m, m', h') \text{ if } h = 0, \\ 1 \text{ if } h' = h - 1 \text{ and } m' = m, \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

$\mathcal{Q}(\langle m, s, h \rangle, a, o) = 1$ if $s = o$ and 0 otherwise, $\mathcal{R}(\langle m, s, h \rangle, a) = R_m(s, a)$.

Discussions. When considering non-stationary environments in MDPs, each component of the quadruplet $\langle \mathbf{S}, \mathbf{A}, T, R \rangle$ may be impacted by an environmental change. Indeed, some states may become impossible or new states may become reachable, some actions may become infeasible or new actions may appear, the transition function and the reward function can of course also change after the environment evolves. Interestingly, a change in the set of states and/or the set actions may always be modeled by a change in the transition and reward functions by considering the set of all possible states for $\mathbf{S}$ and the set of all possible actions for $\mathbf{A}$ at the beginning.

It is easy to notice that HM-MDPs form a subclass of HS3MDPs. In fact, a problem represented as an HS3MDP can also be exactly represented as an HM-MDP by augmenting the modes. The two models are equivalent in the following sense. A model $\mathcal{M}$ is *equivalent* to a model $\mathcal{M}'$ if and only if a problem that can be represented in model $\mathcal{M}$ can also be exactly represented in model $\mathcal{M}'$ and vice-versa.

Proposition 1. *HM-MDPs are equivalent to HS3MDPs.*

Proof. $\Rightarrow$ Given an HM-MDP, we can define an equivalent HS3MDP by setting a mode duration function H such that $\forall m, m', H(m, m', 1) = 1$ and $H(m, m', h) = 0, \forall h \neq 1$. At each timestep, $h = 0$, thus leading only to the first alternative of 3. This turns out to be the exact formulation of an HM-MDP.

$\Leftarrow$ Given an HS3MDP, we show how to build an equivalent HM-MDP. To that aim, we build a sequence of equivalent HS3MDPs. Denote $\langle \mathbf{M}_1, C_1, H_1 \rangle$ the initial HS3MDP. We repeat the following operation to build the sequence: If, for $\langle \mathbf{M}_i, C_i, H_i \rangle$, there exist $m, m' \in \mathbf{M}_i$ and $h \neq 1$ such that $H_i(m, m', h) > 0$, we define the next HS3MDP $\langle \mathbf{M}_{i+1}, C_{i+1}, H_{i+1} \rangle$ as follows:

$$\begin{array}{l} \mathbf{M}_{i+1} = \mathbf{M}_i \cup \bigcup_{h' \neq 1} \{m'_0, \ldots, m'_{h'-1} | H_i(m, m', h') > 0\} \\ C_{i+1}(m, m'_{h'-1}) = C_i(m, m') \times H(m, m', h') \\ C_{i+1}(m'_j, m'_{j-1}) = 1, \forall j > 0 \\ C_{i+1}(m_1, m_2) = C_i(m_1, m_2), \forall (m_1, m_2) \neq (m, m') \\ H_{i+1}(m, m'_{h'-1}, 1) = H_{i+1}(m'_j, m'_{j-1}, 1) = 1, \forall h' > 0, j > 0 \\ H_{i+1}(m_1, m_2, h') = H_i(m_1, m_2, h'), \forall (m_1, m_2) \neq (m, m'), \forall h' \end{array} \tag{4}$$

where for all j, m'_j is a duplicate of m' and C_{i+1} and H_{i+1} are null for the unspecified cases. When this operation cannot be iterated, in the last HS3MDP, unreachable modes can be removed. Finally, the resulting HS3MDP corresponds to an equivalent HM-MDP. □

However, representing HS3MDPs in such a way feels unnatural and leads to a higher number of modes, which moreover, would have a negative impact on the solving time. It is also obvious that, if the maximum duration is unbounded, the equivalent HM-MDP would have an infinite number of modes, making it difficult to solve.

As a final note, the models of HM-MDPs and HS3MDPs are particular instances of Mixed-Observable MDPs (MOMDPs) [12, 1], a subclass of POMDPs. Therefore, MOMDPs algorithms could be used for solving HS3MDPs. We chose to base our solving method on POMCP, because it tends to be more efficient than specialized algorithms on MOMDPs and more generally on factored POMDPs, even when POMCP is run on the non-factored representations [16].

4 Solving an HS3MDP

As HM-MDPs form a subclass of HS3MDPs, solving exactly an HS3MDPs is a PSPACE-complete problem [5]. In order to be able to tackle large instances of problems, we therefore focus on an approximate solving algorithm. A first naive approach is to apply POMCP (see Sect. 2) to directly solve the POMDP derived from an HS3MDP. In that case, a particle in POMCP represents a mode m, a state s and a duration h of the HS3MDP. We propose in this section two possible improvements to this naive approach. Notice that, as a subclass of HS3MDPs, these solving methods can also be applied to HM-MDPs. In the remaining of the paper, we will therefore focus only on HS3MDPs.

Adaptation to the structure. In large instances, POMCP can suffer from a lack of particles to approximate the belief state especially if the number of states in the POMDP and/or the horizon are large. To tackle this issue, a particle reinvigoration technique can be used in the original algorithm. However, it is often insufficient. When POMCP runs out of particles, it samples the action set according to a uniform distribution, which obviously leads to suboptimal decisions.

We propose a first adaptation of POMCP that exploits the structure of HS3MDPs to postpone the lack of particles. In fact, in the derived POMDP, as the agent observes a part of the state of the POMDP, a particle needs only to represent non-observable information, that is, the mode m and the number of steps to stay h. This adaptation allows us to initially distribute particles over a set whose cardinality is much smaller. However, the size of the particle set $|B(\tau)|$ still depends on the number of simulations. This modification of POMCP is introduced at line 3 of Alg. 1.

Exact representation of the belief state. When solving large-sized problems, the above adaptation of POMCP still suffers from lack of particles. We thus

propose a second adaptation where we replace the particle set B by an exact representation of the belief state. This representation consists of a probability distribution μ over $\mathbf{M} \times \mathbb{N}$ (modes and duration in the current mode).

Lines 3 and 5 of Alg. 1 are modified as particles are now drawn according to a probability distribution. Line 22 is not needed anymore. This probability distribution is updated after a new observation using the following equation:

$$\mu'(m', h') = \frac{1}{K}\big(T_{m'}(s, a, s') \times \mu(m', h' + 1) + \sum_{m \in \mathbf{M}} C(m, m') \times T_m(s, a, s') \times \mu(m, 0) \times H(m, m', h' + 1)\big) \ . \tag{5}$$

where K is the normalization term and elements s, s', a are respectively the previous observation, the new observation given by the real environment and the action performed and given by the procedure SEARCH. This update is performed after every action executed in the real environment.

In HM-MDPs we can rewrite the above equation knowing $\mu(m', h' + 1) = 0, \forall m', h'$ and $H(m, m', 1) = 1$. We then obtain:

$$\mu'(m') = \frac{1}{K}\big(\sum_{m \in \mathbf{M}} C(m, m') \times T_m(s, a, s') \times \mu(m)\big) \ . \tag{6}$$

We fall back to the HM-MDP update equation described by [8].

Unlike the previous adaptation, the spatial complexity of this one does not depend on the number of simulations. Indeed, μ is a probability distribution over $\mathbf{M} \times \mathbb{N}$. Assuming a finite maximum number $h_{\max}$ of timesteps to stay in a mode, which is often the case in practice, there always exists a number of simulations N for which the size of the particle set is greater than the length of this distribution. In such a case, this second adaptation will be more interesting to consider. The time complexity of the update of the exact representation is $\mathcal{O}(|\mathbf{M}| \times h_{\max})$. It is to be compared to the particle invigoration of the original POMCP and the first adaption which is $\mathcal{O}(N)$ with N being the number of simulations.

5 Experimental Results

We tested POMCP and our two adapted versions on four non-stationary problems. The first three environments are problems of the literature [8]. We solved an extended version of each problem modeled as an HS3MDP. Recall that those adapted versions of the problems cannot be represented as efficiently with HM-MDPs (see Prop. 1). Results for this model are thus not reported.

$H(m, m', \cdot)$ is defined as a truncated Gaussian probability distribution on duration h of the mode m' after a transition from m. The mean of the Gaussian is uniform randomly drawn between 1 and 5 when creating the environment.

We present the results for the original POMCP and for our adaptations of POMCP: the Structure Adapted (SA) and Structure Adapted combined with

the Exact Representation (SAER) of belief states. We also show the results of the optimal policy when it could be computed, using *Finite Grid* [4] and *MO-IP* [1]. We also used *MO-SARSOP* [12] with one hour of policy computation time when the model could be generated for offline computing. We present the performances of the algorithms for several numbers of simulations to study how the quality of the solutions evolves. For each number of simulations we averaged the cumulative discounted rewards over 1000 runs. We reported results that could be obtained within one hour on a computer equipped with an Intel XeonX5690 4.47 Ghz core. We chose to present the raw results for the original POMCP and percentages for the others. Reported percentages correspond to the improvement in the average cumulated discounted rewards between our modified versions and the original POMCP.

Traffic light. In the traffic light problem, the environment is a two-way road where the system has to choose which side to let pass. It has to decide which traffic light to switch on, knowing only the current state of the lights and the presence or not of cars on each side of the road. In this problem, the HS3MDP has two modes: rush on the left or on the right and two actions to choose which light to switch on. The model contains eight states depending on the presence or not of cars on the left, on the right and on the light state. The reward function gives a negative reward when a car waits on a side of the road whose light is shut off. At each timestep, the environment has a probability of 0.9 to stay in the same mode and 0.1 to change. Finally, the transition function over the state depends on the probability of cars arriving on each side, according to the current mode. Exact probabilities for the original problem can be found in [6].

Table 1 describes results for the traffic light problem, using different algorithms: original POMCP (orig.), Structure Adapted (SA), Structure Adapted combined with Exact Representation of belief states (SAER) and Finite Grid, MO-IP and MO-SARSOP. The last three algorithms yield the same results, which are presented in column "Opt." to give an idea of the optimal value. The performances of the original POMCP almost strictly increase with the number of simulations. They therefore get closer to the optimal value, which translates into decreasing percentages in Column "Opt." of Table 1. Since our modified versions of POMCP performs better than the original one (positive percentages for columns "SA" and "SAER"), they also get closer to the optimal. For instance, with 512 simulations, 4.7% of improvement for SAER compared to 9.3% for Column "Opt." means that the performances of SAER are half-way between those of the original POMCP and the optimal value. Note that a decreasing percentage does not mean a raw decrease in the performances. It means that the increase of the performances of the original POMCP is higher than those of the other algorithms. Nonetheless, the percentages being positive, the later still perform better.

Theoretically, POMCP converges towards the optimal solution while the number of simulations increases. Experimental results (Table 1) show that it

Sim.	Orig.	SA	SAER	Opt.
1	-3,42	0.0%	0.0%	38.5%
2	-2,86	3.0%	**4.0%**	26.5%
4	-2,80	8.1%	**8.8%**	25.0%
8	-2,68	6.0%	**9.4%**	21.7%
16	-2,60	**8.0%**	**8.0%**	19.2%
32	-2,45	5.3%	**6.9%**	14.3%
64	-2,47	**10.0%**	9.1%	14.9%
128	-2,34	**4.3%**	3.4%	10.4%
256	-2,41	8.5%	**10.5%**	12.7%
512	-2,32	**5.6%**	4.7%	9.3%
1024	-2,31	5.1%	**7.0%**	9.3%
2048	-2,38	9.0%	**10.5%**	11.8%

Table 1: Results for traffic light

Sim.	Orig.	SA	SAER	MO
1	60	**11.7%**	6.7%	408.3%
2	63	**30.2%**	**30.2%**	384.1%
4	55	38.2%	**54.5%**	454.5%
8	70	8.6%	**27.1%**	335.7%
16	59	13.6%	**88.1%**	416.9%
32	66	28.8%	**92.4%**	362.1%
64	90	21.1%	**45.6%**	238.9%
128	94	53.2%	**71.3%**	224.5%
256	119	48.7%	**76.5%**	156.3%
512	159	**31.4%**	27.0%	91.8%
1024	177	20.9%	**28.8%**	72.3%
2048	206	**13.6%**	10.2%	48.1%
4096	226	12.4%	**16.4%**	35.0%
8192	227	20.7%	**25.6%**	34.4%

Table 2: Results for sailboat (7×7 grid)

is also the case for our adapted versions whose performances are always at least as good as the original POMCP.

In the traffic light problem, both adaptations of POMCP are roughly even. In fact, the size of the problem is quite small so the original POMCP and the structured adapted POMCP do not lack particles. Moreover, there are enough particles to draw a high quality estimation of the belief state. That is why, the exact representation of belief states does not significantly outperform other POMCP versions. Nonetheless, our adaptations of POMCP both outperform the original version since exploiting the structure of the HS3MDP leads to more accurate belief states.

Sailboat. The sailboat problem is about controlling a boat from a corner of a finite grid to the opposite corner. The states are possible positions in the grid and the modes are the different wind directions, limited to North, South, West and East. Two possible actions manage the sail orientation between North-South and East-West. The transition function over states depends on the sail orientation given the wind direction. The environment has a probability of 0.5 to stay in the same mode, 0.2 to go to an adjacent one and 0.1 to go to the opposite one. The reward function gives a reward of 1 when the goal is reached. This problem can be enlarged as needed by increasing the size of the grid. Results for a 7×7 grid are reported in Table 2.

Due to probabilities of transition between modes, the environment can stay several steps in the same mode thus giving the same wind direction. When the boat is on an edge of the grid, it cannot move until the wind changes to a more favorable configuration. This particularity of the environment leads to a big set of runs where the boat cannot reach the goal and gets stuck on an edge until the end of the run. Moreover, the small drops in the original POMCP performances can be explained with the low number of simulations. If this number is not

Sim.	Orig.	SA	SAER
1	-10.56	0.0%	**1.1%**
2	-10.60	0.0%	0.0%
4	-10.50	2.2%	**3.6%**
8	-10.49	**4.2%**	3.9%
16	-10.44	**5.2%**	5.0%
32	-10.54	**6.2%**	**6.2%**

Table 3: Results for elevator ($f = 7$, $e = 1$)

Sim.	Orig.	SA	SAER
1	-7.41	**1.0%**	0.4%
2	-7.35	**0.3%**	0.0%
4	-7.44	**1.5%**	1.3%
8	-7.35	**0.4%**	0.0%
16	-7.30	**19.1%**	17.2%
32	-7.25	**22.1%**	21.6%
64	-7.17	**24.3%**	**24.3%**
128	-7.22	**27.0%**	**27.0%**

Table 4: Results for elevator ($f = 4$, $e = 2$)

high enough to explore efficiently, the impact of the random can lead to a high variance. Results show that our adaptations always perform better than the original method and that SAER performs almost always better than SA. Column "MO" stands for the results of MO-SARSOP. We can see that SAER converges toward those results as the number of simulations increases.

Elevators. In the elevator problem, the environment can stay in the current mode (see Example 1) with a probability of 0.1 and has a probability 0.45 to change to the other two when the duration is null. Table 3 contains results for an instance with 7 floors and 1 elevator whereas Table 4 shows results for a 4 floors and 2 elevators instance. We were not able to compute the optimal policy for these instances because of their high dimensionality. The results of our adaptations are roughly even since the size of the problem remains quite limited and does not lead to lack particles. However, it is important to note that our methods always outperform the original POMCP whose performances increase with the number of simulations and converge to the optimal solution.

The low number of simulations reached during the computation time is explained by the representation of the transition function. In this problem, transitions are not represented by a matrix of probabilities because of the high number of state components. The transitions are based on a set of rules, leading to a longer computation time.

Randomly generated environments. These environments allow us to study in a controlled setting the scalability of our algorithms. To create an instance, a number of states n_s, actions n_a and modes n_m have to be defined. Random MDPs are then automatically generated such that, in each state, each action can lead to $\lfloor \frac{|\mathbf{S}|}{10} \rfloor$ states and $\lfloor \frac{|\mathbf{S}|}{5} \rfloor$ states can yield a positive reward. To enlarge those environments, we varied the size of the sets of states, actions and modes. We averaged results from 10 different instances with different state/mode transition and reward functions for each parameter set.

Tables 5, 6 and 7 describe results for randomly generated environments with respectively 5, 10 and 20 modes. We were not able to compute the optimal policy for these instances because of their high dimensionality. We can see that our

Sim.	Orig.	SA	SAER
1	0.41	0.0%	**5.6%**
2	0.41	4.9%	**51.4%**
4	0.42	11.5%	**140.9%**
8	0.44	30.9%	**209.6%**
16	0.48	34.6%	**234.7%**
32	0.58	46.0%	**223.0%**
64	0.77	53.1%	**187.2%**
128	1.08	45.7%	**123.4%**
256	1.52	33.5%	**70.0%**
512	1.98	19.6%	**34.5%**
1024	2.30	12.5%	**17.3%**

Table 5: Results for random environments with $n_s = 50$, $n_a = 5$ and $n_m = 5$

Sim.	Orig.	SA	SAER
1	0.39	0.1%	**8.9%**
2	0.39	21.0%	**57.5%**
4	0.40	9.9%	**149.0%**
8	0.41	24.0%	**224.6%**
16	0.43	33.0%	**261.3%**
32	0.48	58.2%	**275.8%**
64	0.60	76.2%	**248.7%**
128	0.83	75.4%	**184.5%**
256	1.16	64.1%	**115.9%**
512	1.61	41.5%	**61.5%**
1024	2.05	2.2%	**28.8%**

Table 6: Results for random environments with $n_s = 50$, $n_a = 5$ and $n_m = 10$

Sim.	Orig.	SA	SAER
1	0.39	0.8%	**11.9%**
2	0.40	2.6%	**51.1%**
4	0.40	2.7%	**138.9%**
8	0.41	11.8%	**225.2%**
16	0.41	22.3%	**270.8%**
32	0.45	42.9%	**290.3%**
64	0.51	77.5%	**305.5%**
128	0.63	102.2%	**261.1%**
256	0.85	102.7%	**186.8%**
512	1.23	73.3%	**107.7%**
1024	1.66	43.6%	**55.3%**

Table 7: Results for random environments with $n_s = 50$, $n_a = 5$ and $n_m = 20$

methods significantly outperform the original POMCP method. In fact, the exact representation of belief states always outperforms POMCP versions based on particles filter on sufficiently large environments. Indeed, these methods quickly lack particles to accurately represent the belief state.

Moreover, the computation time of our adaptations are promising for application to large-sized real life problems. For instance, in the random environment with 20 modes (Table 7), one run of 1024 simulations took 1.15 seconds for solving the HS3MDP with structured adapted POMCP and 1.48 seconds for solving the HS3MDP with POMCP and exact representation of the belief state.

6 Conclusion and discussions

In this paper, we introduced Hidden-Semi-Markov-Mode Markov Decision Processes (HS3MDPs), a new generalization of Hidden-Mode Markov Decision Processes (HM-MDPs) to handle in a more natural and efficient way non-stationary environments. We proposed to use the Partially Observable Monte-Carlo Planning algorithm as a solving method for HS3MDPs. As a subclass of our model, HM-MDPs can be solved efficiently using the same methods. However, this algorithm does not solve large-sized problems modeled with HS3MDPs in the most efficient way. We developed two adaptations of POMCP to improve its performances. The first adaptation exploits the structure of HS3MDPs to alleviate particle deprivation. The second adaptation uses an exact representation of the belief state to reach better results with less simulations than the other two methods. Experimental results on various domains of the literature show that those adaptation significantly improve the performance.

As future work, different research directions could be explored. In HM-MDPs and HS3MDPs, transition functions over modes do not depend on the performed

action. This assumption does not hold in environments like stock markets where buying big volumes may influence the market. An extension of HS3MDPs to handle such situations would be interesting. Another research direction is to relax the assumption that the transition function between modes is known and to learn it in a multi-armed bandit or reinforcement learning setting.

Acknowledgments

Funded by the French National Research Agency under grant ANR-10-BLAN-0215.

References

1. Araya-López, M., Thomas, V., Buffet, O., Charpillet, F.: A closer look at MOMDPs. In: ICTAI (2010)
2. Aström, K.: Optimal control of markov decision processes with incomplete state estimation. J. of Math. Analysis and Applications 10, 174–205 (1965)
3. Cassandra, A., Littman, M., Zhang, N.: Incremental Pruning: A simple, fast, exact method for Partially Observable Markov Decision Processes. In: UAI. pp. 54–61 (1997)
4. Cassandra, T.: pomdp-solve. http://www.pomdp.org/code/index.shtml/ (2003–2013)
5. Chadès, I., Carwardine, J., Martin, T.G., Nicol, S., Sabbadin, R., Buffet, O.: MOMDPs: A solution for modelling adaptive management problems. In: AAAI (2012)
6. Choi, S.: Reinforcement learning in nonstationary environments. Ph.D. thesis, Hong Kong Univ. of Science and Tech. (2000)
7. Choi, S., Yeung, D., Zhang, N.: An environment model for nonstationary reinforcement learning. In: NIPS. pp. 981–993 (2000)
8. Choi, S., Zhang, N., Yeung, D.: Solving Hidden-Mode Markov Decision Problems. In: AISTATS. pp. 19–26 (2001)
9. Doya, K., Samejima, K., Katagiri, K., Kawato, M.: Multiple model-based reinforcement learning. Neural Computing 14(6), 1347–1369 (2002)
10. Kaelbling, L., Littman, M., Cassandra, A.: Planning and acting in partially observable stochastic domains. Artificial intelligence 101(1–2), 99–134 (1998)
11. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo planning. In: European Conference on Machine Learning (2006)
12. Ong, S., Png, S., Hsu, D., Lee, W.: POMDPs for robotic tasks with mixed observability. In: Robotics: Science & Syst. (2009)
13. Papadimitriou, C., Tsitsiklis, J.: The complexity of Markov Decision Processes. Math. of OR 12(3), 441–450 (1987)
14. Puterman, M.: Markov Decision Processes: Discrete dynamic stochastic programming. John Wiley Chichester (1994)
15. da Silva, B., Basso, E., Bazzan, A., Engel, P.: Dealing with non-stationary environments using context detection. In: ICML (2006)
16. Silver, D., Veness, J.: Monte-Carlo planning in large POMDPs. In: NIPS. pp. 2164–2172 (2010)
17. Yu, S.: Hidden Semi-Markov Models. Artificial Intelligence 174(2), 215–243 (2010)