A Closer Look at MOMDPs

Mauricio Araya-López*, Vincent Thomas*, Olivier Buffet†, François Charpillet†
*Nancy Université / †INRIA
LORIA – Campus Scientifique – BP 239
54506 Vandoeuvre-lès-Nancy Cedex – France
Email: firstname.lastname@loria.fr – http://maia.loria.fr

***Abstract*—The difficulties encountered in sequential decision-making problems under uncertainty are often linked to the large size of the state space. Exploiting the structure of the problem, for example by employing a factored representation, is usually an efficient approach but, in the case of partially observable Markov decision processes, the fact that some state variables may be visible has not been sufficiently appreciated. In this article, we present a complementary analysis and discussion about MOMDPs, a formalism that exploits the fact that the state space may be factored in one visible part and one hidden part. Starting from a POMDP description, we dig into the structure of the belief update, value function, and the consequences in value iteration, specifically how classical algorithms can be adapted to this factorization, and demonstrate the resulting benefits through an empirical evaluation.**

I. Introduction

Sequential decision-making under uncertainty is an important research field. There has been extensive work on Markov Decision Processes (MDPs) [1] and variants such as Partially Observable MDPs (POMDPs) [2]. Often the difficulties are linked to the size of the state (and action) space, which typically suffers from a combinatorial explosion when increasing the problem size. Then, possible approaches often rely on exploiting the structure of the problem at hand, e.g., by using appropriate heuristics or function approximators, or, if possible, an exact compact representation.

Here, we focus on partially observable Markov decision processes, which are all the more important that numerous real-world decision problems are made difficult by imperfect knowledge about the state of the system at hand (due to partial and noisy observations), e.g., medical diagnosis, surveillance, or machine maintenance [3]. As for MDPs, the most common resolution techniques rely on Dynamic Programming, which implies computing the optimal expected value from each (belief) state, as in the Witness or Incremental Pruning algorithms [4]. Also, a number of advanced approaches rely on the fact that problems often exhibit some structure and can be efficiently modeled as factored POMDPs (fPOMDPs) [5], where states and observations are represented by multiple random variables.

In this paper, we strengthen the analysis of Mixed Observability MDPs (MOMDPs) [6], a formalism that exploits an important property satisfied by many problems: that usually part of the state is "fully" observable. This means that the problem is in-between MDPs and "classical" POMDPs, which can be exploited to reduce the dimensionality of the value function being computed, and therefore speed up various existing algorithms.

In general terms, Zhang and Zhang's *informative* POMDPs [7] exploit a similar idea of reducing the dimensionality of the value function by using information from the observation function to restrict the belief space at each time step. Hsu et al. [8] present a theoretical discussion about fully observed state variables, but MOMDPs where formally presented only recently by Ong et al. [6]. We have independently developed the same idea with different notations, and we present here an improved analysis through a closer look at the consequences of MOMDPs starting directly from the POMDP model.

After presenting background knowledge on POMDPs in Section II, Section III introduces the MOMDP model and shows how it makes it possible to adapt algorithms—in particular Incremental Pruning in Section IV—and make them more efficient. The benefit of this approach is then evaluated empirically in Section V before a discussion and conclusion.

II. Background on POMDPs

POMDPs are usually defined [9] by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, r, b_0 \rangle$ where, at any time step, the system being in some state $s \in \mathcal{S}$ (the *state space*), the agent performs an action $a \in \mathcal{A}$ (the *action space*) that results in (1) a transition to a state s' according to the *transition function* $T(s,a,s') = Pr(s'|s,a)$, (2) an observation $o \in \mathcal{O}$ (the *observation space*) according to the *observation function* $O(s',a,o) = Pr(o|s',a)$ and (3) a scalar *reward* $r(s,a)$. b_0 is the initial probability distribution over states. Unless stated otherwise, the state, action and observation sets are finite [4].

The problem is for the agent to find a decision *policy* π choosing, at each time step, the best action based on its past observations and actions to maximize its future gain (which can be measured for example through the total accumulated reward or the average reward per time step). Compared to classical deterministic planning, the agent has to face the difficulty to account for a system not only with uncertain dynamics but also whose current state is imperfectly known.

The agent can typically reason about the hidden state of the system by computing a *belief state* $b \in \mathcal{B} = \Pi(\mathcal{S})$ (the set of probability distributions over $\mathcal{S}$) using the following update formula (based on the Bayes rule) when performing action a and observing o:

$$b^{a,o}(s') = \frac{O(s',a,o)}{Pr(o|a,b)} \sum_{s\in\mathcal{S}} T(s,a,s')b(s),$$

where $Pr(o|a,b) = \sum_{s,s''\in\mathcal{S}} O(s'',a,o)T(s,a,s'')b(s)$. Using belief states, a POMDP can be rewritten as an MDP over the belief space, or *belief MDP*, $\langle \mathcal{B}, \mathcal{A}, \mathcal{T}, \rho\rangle$, where the new transition and reward functions are both defined over $\mathcal{B}\times\mathcal{A}\times\mathcal{B}$. With this reformulation, a number of theoretical results about MDPs can be extended, such as the existence of a deterministic policy that is optimal. An issue is that, even if a POMDP has a finite number of states, the corresponding belief MDP is defined over a continuous—and thus infinite—belief space.

For now we only consider finite horizon problems ($t \in 0..T$), maximizing the cumulative reward and looking for a policy taking the current belief state as input. The objective is then to find an optimal policy verifying $\pi^* = \arg\max_{\pi\in\mathcal{A}^{\mathcal{B}}} J^\pi(b_0)$ with

$$J^\pi(b_0) = E\left[\sum_{t=0}^{T-1} r_t \middle| b_0, \pi\right],$$

where b_0 is the initial belief state and r_t the reward obtained at time step t. Bellman's principle of optimality [10] lets us compute this function recursively through the *value function*

$$V_n(b) = \max_{a\in\mathcal{A}} \left[\rho(b,a) + \beta \sum_{b'\in\mathcal{B}} \phi(b,a,b')V_{n-1}(b')\right], \quad (1)$$

where, for all $b\in\mathcal{B}$, $V_0(b) = 0$, and $J^\pi(b) = V_{n=T}(b)$.

This recursive computation has the property to generate piecewise-linear and convex (PWLC) value functions for each horizon [2], i.e., each function is determined by a set of hyperplanes (each represented by a vector), the value at a given belief point being that of the highest hyperplane (see Figure 1-b). For example, if Γ_n is the set of vectors representing the value function for horizon n, then $V_n(b) = \max_{\gamma\in\Gamma_n} \sum_s b(s)\gamma(s)$.

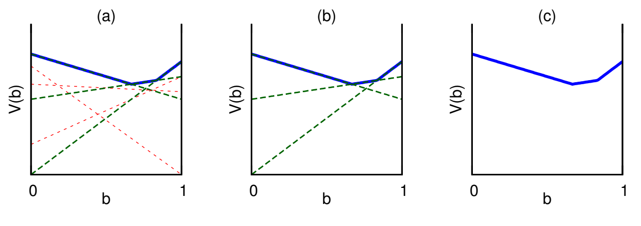

Figure 1. Γ-set representation of a value function in a 1D belief space with (a) and without (b) unnecessary hyperplanes (see Sec. IV). (c) shows the actual value function.

Using the PWLC property, one can perform the Bellman update using the following factorization of Equation 1:

$$V_n(b) = \max_{a\in\mathcal{A}} \sum_o \sum_s b(s) \left[\frac{r(s,a)}{|\mathcal{O}|} + \sum_{s'} T(s,a,s')O(s',a,o)\chi_{n-1}(b^{a,o},s')\right] \quad (2)$$

with $\chi_n(b) = \arg\max_{\gamma\in\Gamma_n} b\cdot\gamma$.[1] If we consider the term in brackets in Equation 2, this generates $|\mathcal{O}|\times|\mathcal{A}|$ Γ-sets, each one of size $|\Gamma_{n-1}|$. These sets are defined as

$$\overline{\Gamma_n}^{a,o} = \{\frac{r^a}{|\mathcal{O}|} + P^{a,o}\cdot\gamma_{n-1}, \forall\gamma_{n-1}\in\Gamma_{n-1}\},$$

where $P^{a,o}(s,s') = T(s,a,s')O(s',a,o)$, $r^a(s) = r(s,a)$.

Yet, these $\overline{\Gamma_n}^{a,o}$ sets are *non-parsimonious*: γ-vectors whose corresponding hyperplanes are below the value function are useless (see Fig. 1-a and b). Pruning phases are then required to remove dominated vectors.

There are several pruning algorithms for exactly solving POMDPs like *Batch Enumeration* [11] or more efficient algorithms such as *Witness* or *Incremental Pruning* [4]. On the other hand, advanced resolution techniques have been developed to tackle the high computational complexity of POMDPs [12], for example by approximating the value function as in *Point-Based Value Iteration* (PBVI) [13], *Heuristic Search Value Iteration* (HSVI) [14], PERSEUS [15] or SARSOP [16].

III. Mixed Observability MDPs

Classical POMDPs are essentially an *indirectly* observable MDPs, because the information about the state is obtained indirectly through instant observations. On the other hand, in an MDP the current state is directly observed at each step. MOMDPs propose a middle-ground scenario, where some of the state variables can be directly observed—namely *visible variables*—and the remaining ones are *hidden variables*.

The rest of this section presents our theoretical results of distinguishing between visible and hidden state variables starting directly from the POMDP formalization, explaining step by step how MOMDPs can be derived. This is a complementary result to Ong et al.'s work [6], where MOMDPs are presented in a practical fashion to solve robotic tasks using approximation algorithms.

A. A Closer Look at the MOMDP Formalization

Factored POMDPs present the state s (and possibly the observation o) as a vector of variables in a view to exploit the underlying structure of the problem as typically done in probabilistic graphical models. Following a similar idea, in MOMDPs the state s is decomposed only into two variables: s_v which is the visible part of the state, and s_h which is the hidden part. This simple distinction triggers several

[1]The χ function returns a vector, so $\chi_n(b,s) = (\chi_n(b))(s)$.

interesting results in the POMDP framework and in their solution techniques.

If we examine more closely the idea of splitting the state into hidden and visible states, we will see that the observation can also be divided in two, as illustrated by Fig. 2 as a dynamic influence diagram (DID) [17]. In this factorization, the visible state variable is redundantly included in both state and observation, because this is the standard way of modeling visible variables in POMDPs. Therefore, the "visible" counterpart of the observation o_v is the same as the visible state s_v. On the other hand, the rest of the observation, o_w, can depend on the whole system state and the last action taken.

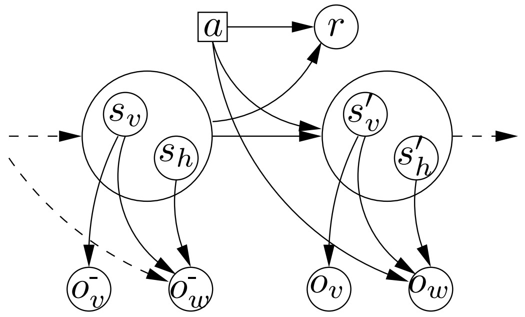

Figure 2. A MOMDP viewed as a dynamic influence diagram

Formally, the state and observation spaces can both be partitioned in two: $\mathcal{S} = \mathcal{S}_v \times \mathcal{S}_h$, $\mathcal{O} = \mathcal{O}_v \times \mathcal{O}_w$. The transition and observation functions both remain the same, but can be expressed in terms of hidden and visible state variables: $T(s, a, s') = T(s_v, s_h, a, s'_v, s'_h)$, $O(s', a, o) = O(s'_v, s'_h, a, o_v, o_w)$. The visible part of the state is duplicated in the observation, so if these parts do not match, then the observation probability goes to zero. This can be formally expressed by dividing the joint probability in two: $Pr(o_v, o_w|a, s'_v, s'_h) = Pr(o_v|a, s'_v, s'_h, o_w)Pr(o_w|a, s'_v, s'_h)$, i.e.:

$$O(s', a, o) = \delta(o_v, s'_v)\Omega(s'_h, s'_v, a, o_w), \tag{3}$$

where $\Omega(s'_h, s'_v, a, o_w) = \sum_{o_v} O(s'_v, s'_h, a, o_v, o_w)$ and $\delta(x, y) = 1$ if $x = y$ and 0 otherwise.

The Ω function is the marginal observation function over the o_w observation given an action a, a hidden state s'_h and a visible state s'_v. Usually, this function can be directly obtained from the problem definition, but if not, it can be computed at very low cost.

B. A Closer Look at the Belief Update

The belief-state for a visible variable will be always a probability distribution completely concentrated (i.e. probability of 1) on the last observed value of that variable. Therefore, a belief-state point in the proposed framework can be represented by a tuple $b = (s^b_v, b_h)$, where s^b_v is the last observed value of the visible state, and b_h the belief-state for the hidden variable. This representation is a fair compression of the same information that a standard belief-state contains for a problem with visible state variables.

As a belief state is a probability distribution over the states, $Pr(s|b)$, the joint distribution $Pr(s_v, s_h|s^b_v, b_h)$ can be written as $Pr(s_v|s^b_v)Pr(s_h|b_h)$, i.e.:

$$b(s) = \delta(s^b_v, s_v)b_h(s_h). \tag{4}$$

Consequently, the belief state update can be done separately for s^b_v and b_h, so that b^{a,o_w,o_v}_{t+1} is computed as follows:

$$\begin{aligned} s^b_{v,t+1} &= o_v \\ b_{h,t+1}(s'_h) &= Pr(s'_h|a, o_w, o_v, s'_v, s^b_{v,t}, b_{h,t}), \forall s'_h \in \mathcal{S}_h \\ &\quad (= Pr(s'_h|a, o, b)), \end{aligned} \tag{5}$$

where the s'_v term can be removed from the belief update of the hidden state, because Equation 3 obliges $s'_v = o_v$.

The presented hybrid representation of a belief state point—half probability distribution and half state value—is the main reason why the MOMDP approach is a middle-ground technique between POMDPs and MDPs. Fig. 3 provides a graphical explanation of this issue. The first image (on the left) shows an (hyper)plane representation of a value function in normal POMDPs, where both the s_v and s_h variables[2] are considered hidden, and are therefore represented by continuous belief state variables b_v and b_h. The third image (on the right) lays out the MDP representation of the same value function, where both variables are directly observable, so that there is no need to summon belief state variables. Here, the value function is represented by a discrete set of points (in this case 4) rather than a set of planes like in the first image. The MOMDP approach is illustrated by the center image, where one of the dimensions is the belief state over s_h like in POMDPs, but the other is directly the s_v variable like in MDPs. With this approach, the belief state over s_h is a continuous variable, but the visible variable's state space (for s_v) is discrete. As a result, the value function is represented as a *set of sets of lower dimensionality hyperplanes*, which can be seen graphically as slices (also called "cuts") of the normal POMDP hyperplanes in Fig. 3.

C. A Closer Look at the Value Function

The hybrid belief state representation and the marginal observation function presented above lead to a more efficient computation of the value function knowing the visible state variables. Specifically, we now present how to compute the value function in terms of a *set* of parsimonious representations of low-dimensional Γ-sets, rather than one unique set as in the classical solution techniques.

The main idea is to use the results of the past sections in order to rewrite Equation 2 in terms of Equations 3, 4

[2]For plotting purposes s_v and s_h are simple binary variables.

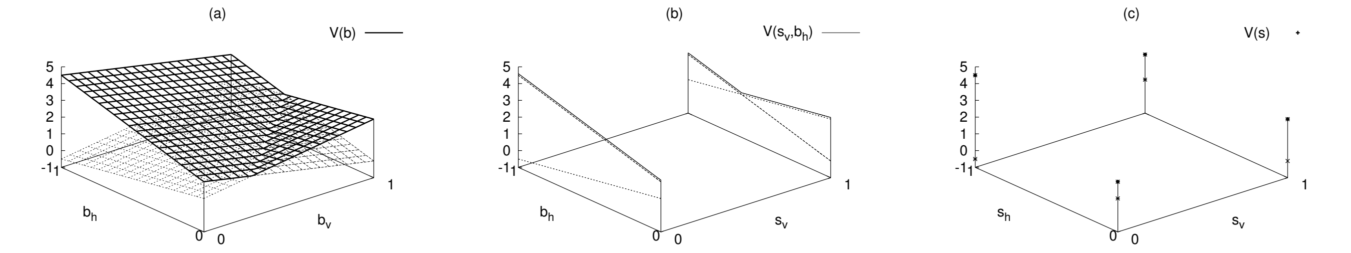

Figure 3. The same value function viewed in a POMDP, a MOMDP, and an MDP. – Note: This is a convenient but abusive representation: a value function over 4 states requires a 4D plot.

and 5, the hybrid representation of $b = (s_v^b, b_h)$, and the basic result $\sum_a \delta(a,b) f(a) = f(b)$:

$$V_n(s_v^b, b_h) = \max_{a \in \mathcal{A}} V_n^a(s_v^b, b_h)$$

$$V_n^a(s_v^b, b_h) = \sum_{o_w} \sum_{o_v} V_n^{a,o_w,o_v}(s_v^b, b_h)$$

$$V_n^{a,o_w,o_v}(s_v^b, b_h) = \sum_{s_h} b_h(s_h) \gamma_n^{a,o_w,o_v,s_v^b,b_h} \tag{6}$$

$$\gamma_n^{a,o_w,o_v,s_v^b,b_h} = \frac{r(s_v^b, s_h, a)}{|\mathcal{O}_w||\mathcal{O}_v|} + \sum_{s'_h} \Omega(s'_h, o_v, a, o_w) \tag{7}$$

$$\times\, T(s_v^b, s_h, a, o_v, s'_h) \chi_{n-1}(b_h^{a,o_w,o_v}, o_v, s'_h)$$

with $$\chi_n(b_h, s_v, s_h) = \arg\max_{\gamma_{s_v} \in \Gamma_n^{s_v}} \gamma_{s_v} \cdot b_h. \tag{8}$$

If for normal POMDPs the value function can be represented, at each time-step, as a set of $|\mathcal{S}|$-dimensional vectors, in the MOMDP approach the same value function can be represented as $|\mathcal{S}_v|$ sets of $|\mathcal{S}_h|$-dimensional vectors. This can be explained considering Equation 8, which is a selector function of a γ-vector from the set $\Gamma_n^{s_v}$. The superscript s_v denotes that for each value iteration there are $|\mathcal{S}_v|$ sets of parsimonious representations, each one representing the optimal value function given the visible state s_v for the belief values of s_h.

This leads to forming a set of Γ-sets, that we will call $\Psi_n = \{\Gamma_n^{s_v} | s_v \in \mathcal{S}\}$, where each $\gamma \in \Gamma_n^{s_v}$ has $|\mathcal{S}_h|$ dimensions. Taking into account the structure of Equation 6, the Ψ_n set is a natural way of expressing the value function in terms of both visible states and belief states of the hidden variables.

Each $\Gamma_n^{s_v}$ of the Ψ_n set is constructed independently in the same fashion as for normal POMDPs. Equation 7 now generates $|\mathcal{O}_w| \times |\mathcal{O}_v| \times |\mathcal{A}|$ non-parsimonious sets that must be cross-summed and pruned to obtain each $\Gamma_n^{s_v}$. It is important to notice that the complete set Ψ_{n-1} is needed to calculate each $\Gamma_n^{s_v} \in \Psi_n$. In the next section we will show the advantages of pruning within the MOMDP framework.

IV. Understanding MOMDPs through IP

In order to study what is the real contribution of modeling visible state variables in the POMDP framework—through the MOMDP formalism—we will use a classical exact algorithm rather than bleeding edge algorithms. This is because new algorithms use approximation techniques that could conceal the real impact of MOMDPs.

Monahan's *Batch Enumeration* algorithm [11] uses the update rule followed by a single pruning phase at each iteration:

$$\Gamma_n = \textsc{prune}\left(\bigcup_a \bigoplus_o \overline{\Gamma_n}^{a,o} \right),$$

where $\bigoplus$ is the cross-sum between Γ-sets. In the same form, Cassandra *et al.*'s *Incremental Pruning* algorithm (IP) [18]—which is computationally more efficient—can be written:

$$\Gamma_n = \textsc{prune}\left(\bigcup_a \textsc{prune}\left(\bigoplus_o \textsc{prune}\left(\overline{\Gamma_n}^{a,o} \right) \right) \right).$$

The computation bottleneck of pruning-based algorithms is the resolution of large linear programs (LPs). Each pruning of a non-parsimonious Γ-set requires to solve an LP with $|\Gamma|$ constraints. Therefore, the size of Γ and the vectors' dimensionality determine the time complexity of an LP. Due to Equation 8, it is clear that MOMDPs work with lower dimensionality vectors, at the cost of having more Γ-sets to prune.

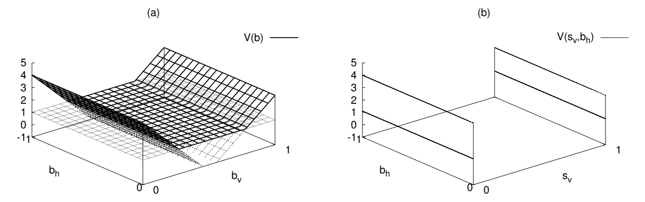

Figure 4. The bottom hyperplane of the POMDP value function (a) disappears in the MOMDP value function (b)

Furthermore, MOMDPs provide more opportunities to prune vectors based on the discrete nature of the visible variables. In Fig. 3-b, the lower vector in the slice $s_v = 0$ is completely dominated, and can be harmlessly removed from the set Γ^0. Fig. 4 shows the case of an horizontal hyperplane that is part of the parsimonious set of a standard POMDP, but is completely removed from all Γ-sets of the MOMDP. This additional vector pruning allows maintaining smaller Γ-sets, and therefore improving the scalability of the algorithms.

With this background, we have enough tools to solve MOMDPs using slight modifications of classical exact POMDP algorithms, in our case Incremental Pruning (IP). Our modifications consist in running one IP-step for each visible state value, starting from a Ψ-set that represents the complete last step value function within the hybrid belief representation. Concretely, the Mixed Observability Incremental Pruning (*MOIP*) extension can be written as

$$\forall s_v \in \mathcal{S}_v, \quad \Gamma_n^{s_v} = \text{PRUNE}\left(\bigcup_a \text{PRUNE}\left(\bigoplus_{o_v}\bigoplus_{o_w} \text{PRUNE}\left(\overline{\Gamma_n}^{a,o_v,o_w,s_v}\right)\right)\right).$$

The solution of the MOIP—or more generally of any mixed-observability-vector-pruning algorithm—is a Ψ_n set of Γ-sets that fully describes the value function. This representation is equivalent to the single Γ-set that the normal IP algorithm returns. A policy graph can be obtained from both, so from end to end MOMDPs are equivalent to—but more efficient than—POMDPs.

Conducting a complexity analysis for a POMDP algorithm being a hard task, we will just mention some important facts. 90–95% of the computation time is spent solving LPs [4], so that we should focus on the number of LPs and on their size. In the worst case, for each LP solved by IP, MOIP solves $|\mathcal{S}_v|$ similar LPs (same number of constraints), this increase being compensated for by the decreased number of variables ($|\mathcal{S}_h|$ instead of $|\mathcal{S}|$). A major issue is to estimate how many hyperplanes disappear by focusing on cuts. This is a difficult question to answer, but a degenerate case that can easily be analyzed is that of point-based algorithms with a grid discretization [19]. The belief space being a $|\mathcal{S}|$-simplex (e.g., a point, a segment, a triangle or a tetrahedron if we have 1, 2, 3 or 4 states), a natural discretization is that which generates a polytopic number of points (see Fig. 5), that is:

$$P_r(n) = \begin{pmatrix} n+r-1 \\ r \end{pmatrix} = \frac{n^{(r)}}{r!},$$

where $n = |\mathcal{S}|$ and r is the resolution (number of points on an "edge", i.e., between two states). When using visible state variables, there is a dramatic saving as this value becomes:

$$p \cdot P_r(m) = p \cdot \begin{pmatrix} m+r-1 \\ r \end{pmatrix} = p \cdot \frac{m^{(r)}}{r!},$$

where $m = |\mathcal{S}_h|$ and $p = |\mathcal{S}_v|$.

V. Understanding MOMDPs through examples

We have implemented a mixed observability version of Incremental Pruning using Cassandra's `pomdp-solve` software.[3] For each experiment several statistics were

[3]http://www.cassandra.org/pomdp/code/index.shtml

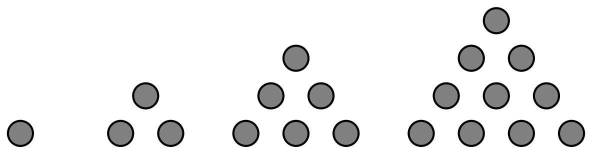

Figure 5. For an n-simplex, the r-th polytopic number $P_r(n)$ is obtained by adding $P_r(n-1)$ to $P_{r-1}(n)$. In the above case of triangular number, this means adding r to T_r. As can be observed, one obtains a discretization of a tetrahedron by stacking up this increasingly large triangles.

gathered—such as time, number of LPs, number of constraints, number of vectors, etc.—but we will present only the two more significant ones: *time* and *solution size*.

Time is the empirical time that Cassandra's solver takes to compute the optimal value function for a given experiment. The time limit for these experiments was 7200 seconds (2 hours).

The *solution size* is the number of elements that a solution file contains. The solution size in the case of the POMDP modeling is $|\Gamma_n| \cdot (|\mathcal{S}| + 1)$, meanwhile in MOMDPs it corresponds to $\sum_{s_v} |\Gamma_n^{s_v}| \cdot (|\mathcal{S}_h| + 1)$. The $+1$ term was included because, for each vector, the solution provides an action value in order to construct the policy.

To investigate the scalability of MOIP compared to IP, experiments have been conducted with two kinds of problems. The first one is the *hide and seek* problem characterized by stochastic transitions and a deterministic observability, the second one is the *lost robot* problem characterized by deterministic transitions but a stochastic observability.

A. The Hide and Seek Problem

This problem involves 2 agents—a hider and a seeker—initially randomly placed on an $n \times m$ grid environment in which b cells are empty and $n.m - b$ cells are walls. The hider has a fixed random behavior. The problem is then for the seeker to maintain an eye contact with the hider, the visibility being computed by a ray-tracing algorithm (see Fig. 6).

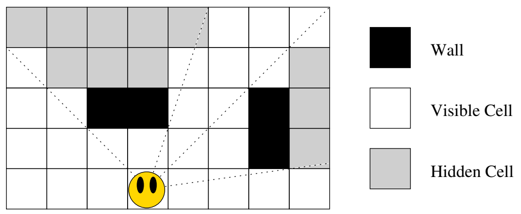

Figure 6. Viewing ability of the seeker agent.

Noting $\mathcal{L} = \{l_1, \ldots, l_b\}$ the set of empty locations, the state is defined by both agents' locations: $s = (l_s, l_h)$, hence $\mathcal{S} = \mathcal{L} \times \mathcal{L}$. Both agents have 9 possible actions corresponding to the 8 chess-king moves plus `stay`, only the seeker's actions being controlled. The instant (deterministic) observation is $o = (l_s, l_h^*)$ where l_s is the seeker's location l_s

and l_h^* is the hider's location if visible, $l_{unknown}$ otherwise ($l_h^* \in \mathcal{L}^* = (\mathcal{L} \cup \{l_{unknown}\})$). The transition, observation and reward functions all depend on the map topology.

This problem is easily modeled as an MOMDP by writing $s = (l_s, l_h) = (s_v, s_h)$ and $o = (l_s, l_h^*) = (o_v, o_w)$. The new observation function is derived trivially.

Experiments Description For this problem we have used *character*-based topologies (letters and numbers, as shown on Fig. 7). These maps are very simple mazes, yet computing the optimal policy for them is not trivial because it depends on the hider's strategy.

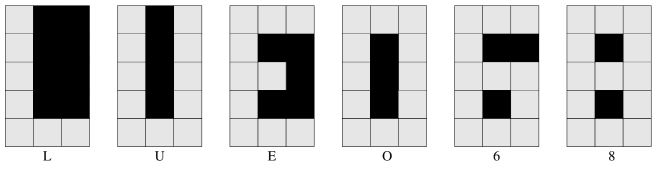

Figure 7. The L, U, E, O, 6 and 8 topologies used to play hide and seek (here on a 3×5 grid).

For these experiments we have used three different strategies for the hider: (1) a *static* strategy where the hider never moves; (2) a *random* strategy, where the hider randomly walks in the maze without considering the seeker position; and (3) an omniscient *reactive* strategy, where the hider deterministically flees from the always known seeker's position. We have analyzed the influence of the horizon, map size, topology complexity and stochastic transition function.

B. The Lost Robot Problem

In this problem, a robot is lost in a toroidal space station made of n identical floors of m rooms each. The floors can be distinguished thanks to the room colors. The goal is to go to the ground level with a minimum number of moves and stop there.

The state of the robot $s = (r, l)$ indicates the current *level* l and *room* r ($r = -1$ if a terminal state has been reached). The current observation $o = (r, c)$ gives the room number r and the (noisy) room color c. The possible actions are to `move` to a neighboring room or floor, or to `stop`. The reward function associates a negative reward r_{move} (-1) for each move, a large positive reward r_{succ} $(+100)$ for a right stop in the ground level and a null r_{fail} reward for a wrong stop. After the stop, the agent receives no more rewards. To obtain the MOMDP model, we simply have to write $s_v = o_v = r$, $s_h = l$ and $o_w = c$.

In these experiments, only circular floors are considered (see Fig. 8), with 2 possible moves: `right` ($r' = (r + 1) \mod m$) and `up` ($l' = (l + 1) \mod n$).

Experiments Description For this problem, we have used two testing sets: (1) randomized two-color maps of different sizes with deterministic observations, and (2) hand-made 2×2 two-color maps with stochastic observations. Because of topology and color symmetries, the second problem set is reduced from 16 to only to 6 different problems (see Fig. 9). With the stochastic observations, there is a 0.9 probability of seeing the right color, and a 0.1 probability of seeing the wrong one. Here, we have analyzed the influence of the map size, topology complexity and stochastic observability.

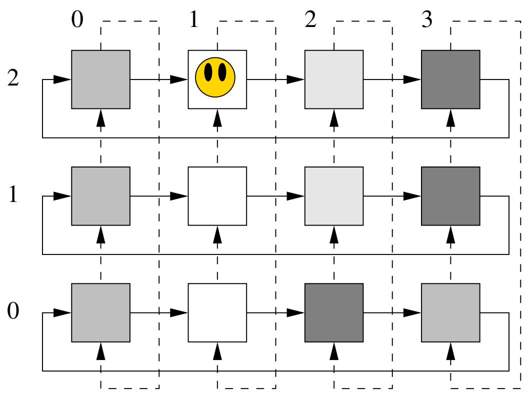

Figure 8. A toroidal space station of size 4×3.

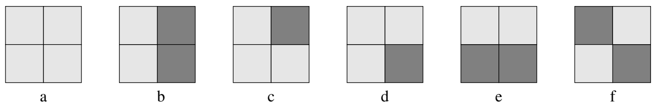

Figure 9. Six 2×2 topologies used for the lost robot problem.

C. Results

Among all the hide and seek problem experiments, we have decided to present only the results with a random strategy for the hider, because the nature of the results is very similar for all the strategies.[4] For the lost robot problem we have used only two-color maps for the same reason.

Table I summarizes some interesting results for both problems. The top part shows time and size for several variations of map topologies and map sizes for the hide and seek problem (Fig. 7). The bottom part—the lost robot problem—presents the whole set of hand-made 2×2 maps (Fig. 9), and the lowest (-l) and highest (-h) speedup values for various sizes of random maps. The horizon for all these experiments was 10, but larger horizons maintain the solution size and mean time per step for each experiment. This is because the optimal γ-vector's structure becomes stable very quickly in problems with small state spaces (3 or 4 steps for most experiments).

It is natural to expect that the solution size increases with the map complexity and size, but the growth of normal IP is considerably faster than for MOIP, as can be seen in the last column ($\times$ |Sol.|). The execution time is also smaller for MOIP in general, and the *speedup* column ($\times$

[4]The results are slightly different in terms of scale and form, but the conclusions about them are the same.

Table I
EXECUTION TIME AND SOLUTION SIZE FOR THE HIDE AND SEEK PROBLEM AND THE LOST ROBOT PROBLEM.

Map	**IP**		**MOIP**		**× factor**	
	Time	|Sol.|	Time	|Sol.|	Time	|Sol.|
Hide and Seek						
L-3x3	0.16	52	0.06	30	2.7	1.7
L-4x4	0.60	100	0.22	56	2.7	1.8
L-5x5	1.71	164	0.60	90	2.9	1.8
L-6x6	4.56	244	1.47	132	3.1	1.9
L-7x7	10.74	340	3.08	182	3.5	1.9
U-3x3	2.71	600	1.06	144	2.6	4.2
U-4x4	40.74	5151	4.18	374	9.8	13.8
U-5x5	246.43	16320	8.91	560	27.7	29.1
U-6x6	1303.40	54227	18.31	816	71.2	66.5
U-7x7	5669.39	153488	36.01	1120	157.4	137.0
O-3x3	587.16	49920	5.62	576	104.5	86.7
E-3x5	–	–	19.26	888	–	–
O-3x5	–	–	80.00	3640	–	–
6-3x5	–	–	197.87	13440	–	–
8-3x5	–	–	687.51	41760	–	–
Lost Robot						
2x2-a	0.01	42	0.01	15	1.0	2.8
2x2-b	0.01	42	0.01	15	1.0	2.8
2x2-c	70.82	4704	0.31	117	228.5	40.2
2x2-d	71.09	4697	0.30	117	237.0	40.2
2x2-e	1355.61	13013	0.96	225	1412.1	57.8
2x2-f	1334.47	12964	0.95	225	1404.7	57.6
3x3-l	0.04	247	0.03	56	1.3	4.4
3x3-h	0.78	1274	0.03	56	26.0	22.8
4x4-l	0.36	1155	0.06	115	6.0	10.0
4x4-h	279.17	5733	0.3	135	930.6	42.5
5x5-l	–	–	0.14	144	–	–
5x5-h	–	–	0.52	720	–	–

Time) shows how many times faster is the MOIP algorithm. For simpler problems MOIP doubles the speed of IP, and when the problem gets more complex (in size, topology or stochastic observability), MOIP behaves even better, with several orders of magnitude improvement at the end.

A good example of the improvement is the O-4×4 case of the hide and seek problem (not shown in Tab. I), where the normal IP did not finish before the two hours even for an horizon of 10, meanwhile MOIP ends up with a solution in a few minutes for an horizon of 100. In the same direction, we explored some cases where the IP algorithm did not finish—E, O, 6 and 8 topologies for hide and seek, and 5x5 random maps for the lost robot—meanwhile MOIP finished in a reasonable time. This shows clearly that MOIP scales much better with the complexity than IP without visible states awareness.

In summary, for all horizons, map topologies, map sizes, stochasticity of the transition function, stochasticity of the observation function, and problem types, MOIP significantly improves over the original IP both in terms of computation time and memory consumption. Also, MOMDPs scale better with the growth of each of the variables we have looked at.

VI. Discussion and Future Work

A difference between Ong et al.'s formulation and ours is that we also present the observation space with a factored representation. The proposed factorization is a simple way of expressing the fact that a part of the state space can be fully disambiguated by any observation value, but the representation is not limited to this one-to-one relationship. In a more general setup, MOMDPs could be extended to represent disambiguation rather than visibility (i.e., s_v may not be identical to part of the observation, but may be unambiguously inferred from o: $s_v = f(o)$), which is a much more generic concept for exploiting the structure of the problem. Nevertheless, the proposed factorization is very useful for the analysis, because it permits converting a POMDP with visible state variables into an MOMDP directly, showing that this new formalism has the same properties as POMDPs, but with a reduction in the time and space complexity. A factored observation space also permits showing the exact shape of the value function (see Eq. 6), which allows applying the MOMDP viewpoint to any technique based on value iteration. Ong et al.'s work is focused specifically on the robotics field, showing that visible state variables are a common property in robotics. However, it is clear that MOMDPs can be applied to other areas, because visible state variables are common in numerous domains. In the same direction, robotic applications are well addressed by fast approximation algorithms (such as SARSOP) and Ong et al.'s results show that they do better with explicit visible state variables. Nevertheless, we show that the MOMDP formalism can be generalized to be applied to other algorithms, and is not restricted to the algorithms presented by Ong et al.

We empirically know that MOMDP versions of Incremental Pruning and SARSOP [16] help decrease time complexity and thus speed up computations. Yet, these algorithms do not benefit from the mixed observability in the exact same way, as we will now see, comparing exact algorithms (as Incremental Pruning), point-based algorithms (as PBVI or SARSOP), and online algorithms (which rely on Monte-Carlo simulations to estimate the action-value function at the current belief state [20]):

- **About belief-point selection:** Let us notice that online algorithms, as well as many point-based algorithms, sample reachable belief points. This means that these classical algorithms naturally focus on the reachable cuts of the belief space. On the contrary, normal (POMDP) point-based algorithms selecting points so as to reduce the error between a lower and an upper-bound will select many non-reachable belief points where not necessary. The latter family of algorithms is therefore more likely to benefit from working on cuts only.
- **About pruning:** MOMDP versions of exact and point-based algorithms both benefit from an efficient pruning

in each cut. Experiments are still missing for MO-SARSOP to measure this phenomenon. This is not relevant for online algorithms as they do not rely on γ-vectors.

It is also important to notice that MOMDPs not only provide an improvement in time complexity, but also in space complexity. The set of γ-vectors, that represent the value function plus their respective action, fully describes the policy to execute. Therefore, MOMDPs usually provide a more compact representation of this policy, as can be seen in Tab. I.

The idea of reducing the dimensionality of the belief space has also been considered in *informative POMDPs* (iPOMDPs) [7]. Here, the starting point is the idea that, given only the last action-observation pair (a, o), the set of possible states $\mathcal{S}^{ao}$ may be of small size. If this is the case for all pairs (a, o), we have an *informative* POMDP, which makes it possible to efficiently work with belief spaces of reduced dimensionality. Considering for example an agent in a maze, looking at the walls surrounding it is sufficient to significantly reduce the number of possible cells it may be standing in. There are notable similarities between iPOMDPs' dimensionality reduction based on the last action-observation pair and MOMDPs' dimensionality reduction through a factorization. Digging the comparison between these approaches or trying to combine them are very promising directions for future work.

VII. Conclusion

In this paper we discussed an important fact about POMDPs: in a number of problems, part of the state is fully observable, so that a large part of the belief space is irrelevant. MOMDPs have been introduced to exploit this fact [6]. We take here a closer look at this formalism, showing how the value function can be modified to account for visible state variables. Our experiments demonstrate a dramatic gain in time complexity of exact algorithms, complementing the results of approximation algorithms presented by Ong et al. As discussed, some algorithms can be adapted to benefit from the MOMDP point of view (e.g. exact and point-based algorithms), whereas others can probably not (e.g. online algorithms). An interesting research direction is to unify MOMDPs with Zhang and Zhang's informative POMDPs and to push them further.

References

[1] D. Bertsekas and J. Tsitsiklis, *Neurodynamic Programming*. Athena Scientific, 1996.

[2] R. Smallwood and E. Sondik, "The optimal control of partially observable Markov decision processes over a finite horizon," *Operation Research*, vol. 21, pp. 1071–1088, 1973.

[3] A. Cassandra, "A survey of POMDP applications," in *AAAI Fall Symposium*, 1998.

[4] ——, "Exact and approximate algorithms for partially observable Markov decision processes," Ph.D. dissertation, Providence, RI, USA, 1998.

[5] E. Hansen and Z. Feng, "Dynamic programming for POMDPs using a factored state representation," in *Proc. of the Int. Conf. on AI Planning and Scheduling*, 2000.

[6] S. Ong, S. Png, D. Hsu, and W. Lee, "POMDPs for robotic tasks with mixed observability," in *Proceedings of Robotics: Science and Systems V (RSS'09)*, 2009.

[7] W. Zhang and N. L. Zhang, "Restricted value iteration: Theory and algorithms," *Journal of Artificial Intelligence Research*, vol. 23, pp. 123–165, 2005.

[8] D. Hsu, W. Lee, and N. Rong, "What makes some POMDP problems easy to approximate?" in *Advances in Neural Processing Systems 21*, 2007.

[9] A. Dutech and B. Scherrer, "Partially observable Markov decision processes," in *Markov Decision Processes in Artificial Intelligence*. ISTE / John Wiley, 2010.

[10] R. Bellman, "The theory of dynamic programming," *Bull. Amer. Math. Soc.*, vol. 60, pp. 503–516, 1954.

[11] G. Monahan, "A survey of partially observable Markov decision processes," *Management Science*, vol. 28, pp. 1–16, 1982.

[12] J. T. C. Papadimitriou, "The complexity of Markov decision processes," *Mathematics of Operations Research*, vol. 12, no. 3, pp. 441–450, 1987.

[13] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence*, 2003.

[14] T. Smith and R. Simmons, "Heuristic search value iteration for POMDPs," in *Proc. of the Int. Conf. on Uncertainty in AI*, 2004.

[15] M. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *Journal of Artificial Intelligence Research*, vol. 24, pp. 195–220, 2005.

[16] H. Kurniawati, D. Hsu, and W. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Robotics: Science and Systems IV*, 2008.

[17] R. Howard and J. Matheson, *Readings on the Principles and Applications of Decision Analysis*, 1984, vol. II, ch. Influence diagrams (1981).

[18] A. Cassandra, M. Littman, and N. Zhang, "Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes," in *Proc. of the 13th Conf. on Uncertainty in AI*, 1997.

[19] R. Zhou and E. Hansen, "An improved grid-based approximation algorithm for POMDPs," in *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence*, 2001.

[20] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online planning algorithms for POMDPs," *Journal of Artificial Intelligence Research (JAIR)*, vol. 32, pp. 663–704, 2008.