Int. J. Appl. Math. Comput. Sci., 2015, Vol. 25, No. 3, 597–615
DOI: 10.1515/amcs-2015-0044

BOTTOM–UP LEARNING OF HIERARCHICAL MODELS IN A CLASS OF DETERMINISTIC POMDP ENVIRONMENTS

HIDEAKI ITOH [a,*], HISAO FUKUMOTO [a], HIROSHI WAKUYA [a], TATSUYA FURUKAWA [a]

[a] Graduate School of Science and Engineering
Saga University, 1 Honjo-machi, Saga 840-8502, Japan
e-mail: {hideaki,fukumoto,wakuya,tach}@ace.ec.saga-u.ac.jp

The theory of partially observable Markov decision processes (POMDPs) is a useful tool for developing various intelligent agents, and learning hierarchical POMDP models is one of the key approaches for building such agents when the environments of the agents are unknown and large. To learn hierarchical models, bottom-up learning methods in which learning takes place in a layer-by-layer manner from the lowest to the highest layer are already extensively used in some research fields such as hidden Markov models and neural networks. However, little attention has been paid to bottom-up approaches for learning POMDP models. In this paper, we present a novel bottom-up learning algorithm for hierarchical POMDP models and prove that, by using this algorithm, a perfect model (i.e., a model that can perfectly predict future observations) can be learned at least in a class of deterministic POMDP environments.

Keywords: partially observable Markov decision processes, hierarchical models, bottom-up learning.

1. Introduction

The theory of partially observable Markov decision processes (POMDPs) provides a general framework for sequential decision making under uncertainty (Drake, 1962; Åström, 1965; Kaelbling *et al.*, 1999). POMDPs have already been used in various problem domains, including robot navigation (Theocharous *et al.*, 2004; Foka and Trahanias, 2007), assistance to elderly people (Pineau *et al.*, 2003; Hoey *et al.*, 2010), and natural language dialog management (Roy *et al.*, 2000; Young *et al.*, 2013). In many studies, however, a model of the environment is given *a priori*. Emphasis has been placed only on how to optimize the policy (i.e., the action-selection rule) for the given model.

The model-based approach is an important branch of POMDP research (Sallans, 2000; Theocharous, 2002; Shani *et al.*, 2005). In this approach, the environment is assumed to be unknown to the agent, and the agent learns the model of the environment through experience (i.e., the history of actions and observations). After learning, the agent optimizes the policy using the model. Given recent advances in policy optimization methods (e.g., Spaan and Vlassis, 2005; Zamani *et al.*, 2012), we can hope that if we have a method for learning the model, the optimal policy can be automatically obtained in the unknown POMDP environment.

Hierarchical models are useful when the environment to be modeled is large (Theocharous and Mahadevan, 2002; Youngblood and Cook, 2007). A hierarchical model typically consists of lower layers, which model some local structures of the environment, and higher layers, which model the global relationships among the local structures. By this mechanism, it can model a large environment with a much smaller number of parameters than flat (i.e., non-hierarchical) models. Therefore, learning can be easier in hierarchical models.

The introduction of *hierarchical POMDPs* (*H-POMDPs*) (Theocharous and Mahadevan, 2002; Theocharous, 2002) and the fast learning method for them (Theocharous *et al.*, 2004) has been an important attempt to enable learning in large POMDP environments. In this learning method, H-POMDPs are represented as dynamic Bayesian networks (DBNs), and these DBNs are efficiently learned via an expectation-maximization (EM) algorithm. This method has been successfully applied to the map learning domain; the models of large environments were more accurately learned using H-POMDPs than using flat POMDPs. However, for this

[*] Corresponding author

method to successfully learn the map, the topology of the map had to be given *a priori*. Thus, learning in large environments without such prior knowledge remains an unsolved problem.

A possible reason why the learning of H-POMDPs has been difficult is that the learning method tries to learn the entire hierarchy simultaneously. A promising alternative approach for learning hierarchical models is the bottom-up approach, in which learning takes place in a layer-by-layer manner from the lowest to the highest layer. In this approach, the lower layers first learn local structures of the environment, and then the higher layers learn the global relationships among the local structures. We expect that this makes learning more convenient, because it would be easier to separately learn the local structures and their relationships than to learn the entire hierarchy simultaneously. Although we could consider top-down approaches (where learning begins at the highest layer), they are difficult to implement because it is hard to learn the relationships among local structures prior to learning the actual local structures.

Indeed, in related research fields, the bottom-up approach has been successfully used in some influential studies, including those of *layered hidden Markov models* (*layered HMMs*) (Oliver *et al.*, 2004) in the HMM literature and *deep learning* (Hinton *et al.*, 2006) in the neural network literature. In logic network structure learning (Mihalkova and Mooney, 2007), the bottom-up approach has been empirically found to be better than top-down approaches.

In the POMDP literature, however, little attention has been paid to the bottom-up approach. A notable exception is the bottom-up learning method that uses data mining techniques (Youngblood *et al.*, 2005; Youngblood and Cook, 2007). This method has been successfully applied to the intelligent home environment domain. However, the method learns hierarchical HMMs (HHMMs) (Fine *et al.*, 1998) and converts them to H-POMDPs. Since this conversion is not always possible, hierarchical POMDP models cannot always be obtained using this method.

Therefore, in this paper, we present a novel bottom-up learning algorithm that directly learns hierarchical POMDP models. We use layers of finite state machines (FSMs) instead of H-POMDPs.

Furthermore, we prove that, by using this algorithm, a perfect model (i.e., a model that can perfectly predict future observations) can be learned in a certain class of POMDP environments. The fundamental idea behind the algorithm is that, as each layer is learning, the learning layer ignores some unimportant parts of the training data (i.e., the history of actions and observations). More specifically, in Section 4.2, we prove that the learning layer can ignore the time periods of the training data during which full observability is recovered by the lower layers.

We also provide empirical results by which we demonstrate how hierarchical models are successfully learned by the algorithm. We show that, in the problem domain used for our experiments, many parts (up to 99.2%) of the training data can be safely ignored when the upper layers are learning. Further, we show that our algorithm outperformed the DBN-based one for learning H-POMDPs (Theocharous *et al.*, 2004) in at least this problem domain.

Thus, the presented algorithm is the first bottom-up algorithm for directly learning POMDP models that comes with a theoretical guarantee and empirical validations regarding its performance.

In this paper, we consider only a subclass of POMDPs. More specifically, the two constraints described below are imposed on the environment. The first constraint is that the environment needs to be a deterministic POMDP (Littman, 1996; Bonet, 2009), in which both state transitions and observations are governed by deterministic functions. Deterministic POMDPs are less general than standard POMDPs in which state transitions and observations are determined by probabilistic functions; however, since the state is hidden from the agent, learning deterministic POMDPs remains a difficult task (Holmes and Isbell Jr., 2006). The second constraint is that the environment cannot have a specific type of loop structure, which we call *action-observation loops* (or *ao-loops* for short); they are defined in Section 3.1. Although this constraint also makes the target environment less general, environments without ao-loops still include non-trivial problems (as we present in Section 5). These two constraints are imposed to simplify the analysis in this paper; it is desired that these constraints are removed in future studies. This study lays a firm foundation for future improvements towards efficient methods for learning hierarchical models in general POMDP environments.

The rest of the paper is organized as follows. Related work is discussed in Section 2. After describing the problem setup in Section 3, we present our learning algorithm in Section 4, which also includes a proof that our algorithm can learn a perfect model. Results of empirical studies are given in Section 5. Section 6 concludes this paper.

2. Related work

There have been various kinds of methods proposed for learning flat POMDP models, e.g., FSM-based methods (Dean *et al.*, 1995; Gavaldà *et al.*, 2006), Bayesian reinforcement learning methods (Poupart and Vlassis, 2008; Ross *et al.*, 2011; Chatzis and Kosmopoulos, 2014), and sample-based methods (Doshi-Velez, 2009; Doshi-Velez *et al.*, 2015). These are different from our approach, since they do not construct hierarchical models.

However, they can be useful for learning each layer of a hierarchical model. For example, Bayesian reinforcement learning methods can be used to optimally explore the partially observable environments. The exploration issue is also important to our approach, although it is not the focus of the present paper; in our experiments, we used a simple policy that selects a random action at any time while the agent learns the environment.

Our hierarchical model is a multi-layer one (e.g., Chang *et al.*, 2003), and each layer can be seen as a kind of jump process (Kołodziej *et al.*, 2011; Rao and Teh, 2013) from the viewpoint of the higher layers, in the sense that the higher layers only occasionally interact with the lower layer. Indeed, our model is closely related to partially observable semi-Markov decision processes (POSMDPs) (White, 1976; Mahadevan, 1998; Lim *et al.*, 2011). A POSMDP is a semi-Markov model (e.g., Li *et al.*, 2007; Oniszczuk, 2009; Rusek *et al.*, 2014) equipped with actions and observations. The relationships between our model and POSMDPs are described in Section 3.3.3.

Hierarchical models have also been extensively studied in the field of hierarchical HMMs (HHMMs) (Fine *et al.*, 1998; Bui *et al.*, 2004; Heller *et al.*, 2009). HHMMs are different from our model because they do not take actions into account. An important attempt to add actions to HHMMs has led to H-POMDPs (Theocharous and Mahadevan, 2002; Theocharous, 2002; Theocharous *et al.*, 2004). We compare our model with this technique in Section 5.

In the reinforcement learning literature, many studies describe model-free approaches for obtaining hierarchical controllers (Dietterich, 2000; Barto and Mahadevan, 2003; Hengst, 2011), mostly in fully observable MDP environments (Kolobov, 2012). These approaches differ from ours, because they do not construct an explicit model of the environment. Comparisons between model-based and model-free approaches are important, but are out of the scope of this paper.

Studies of automatically discovering hierarchical POMDP controllers (Charlin *et al.*, 2007; Toussaint *et al.*, 2008) have some similarities with our study, because they use hierarchical FSMs. However, they are different from our study, because they assume that a model of the environment is given.

3. Problem setup

In this section, we define the environment to be learned, the goal of the learning, and the agent that learns the environment.

3.1. Environment. Let the environment be a deterministic POMDP defined as a tuple $E := \langle S, A, O, \delta, \gamma, s_0\rangle$, where

- $S := \{1, 2, \dots, |S|\}$ is a finite set of states,
- $A := \{1, 2, \dots, |A|\}$ is a finite set of actions,
- $O := \{1, 2, \dots, |O|\}$ is a finite set of observations,
- $\delta : S \times A \to S$ is the deterministic transition function,
- $\gamma : S \times A \to O$ is the deterministic observation function, and
- $s_0 \in S$ is the initial state of the environment.

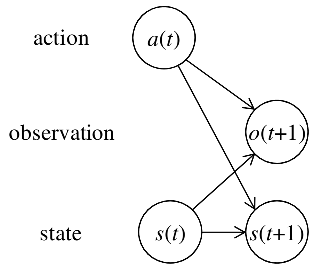


Fig. 1. Time slice of the POMDP environment.

As shown in Fig. 1, at each time $t = 0, 1, 2, \dots$, the agent takes action $a(t) \in A$. Then, state $s(t) \in S$ is changed to $s(t+1) = \delta(s(t), a(t))$, and observation $o(t+1) = \gamma(s(t), a(t))$ is given to the agent. For simplicity, we do not consider *rewards* (i.e., evaluations of the states) in this paper.

We assume that A and O of E are known to the agent; the other elements of E are assumed to be unknown.

Our algorithm is proven to work when the environment does not contain a specific type of structure, which we refer to as *action-observation loops* (or *ao-loops* for short). Before defining ao-loops, we first define the following two fundamental notions.

Definition 1. (*Action-observation sequence*) For any integer n (≥ 1), we call $q = (a_1, o_1, a_2, o_2, \dots, a_n, o_n)$ an *action-observation sequence* if $a_i \in A$ and $o_i \in O$ for all $i \in \{1, 2, \dots, n\}$.

Definition 2. ($\to_q$) Let $q = (a_1, o_1, a_2, o_2, \dots, a_n, o_n)$ be an action-observation sequence, and let s and s' be two (possibly identical) states in S. We write $\boldsymbol{s \to_q s'}$ if, by receiving the action sequence specified in q (i.e., $(a_1, a_2, \dots, a_n)$), the state s transitions to s' emitting the observation sequence specified in q (i.e., $(o_1, o_2, \dots, o_n)$).

Given these definitions, we can define ao-loops. We divide our definition into what we call *1st-order ao-loops* (specified as Definition 3 and shown in Fig. 2(a)) and *n-th-order ao-loops* (specified as Definition 4, and two examples shown in Figs. 2(b) and 2(c)).

Definition 3. (*1st-order ao-loop*) In environment E, if there exist two different states s and s' and one action-observation sequence q such that there is a transition path $s \to_q s' \to_q s'$, then we call this path a *1st-order ao-loop*.

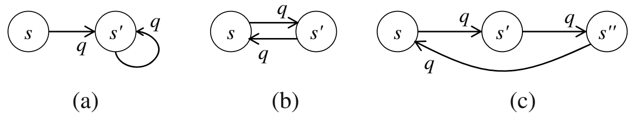


Fig. 2. Examples of ao-loops: 1st-order ao-loop, where s and s' are two different states (a), 2nd-order ao-loop, where s and s' are two different states, too (b), and 3rd-order ao-loop, where s, s', and s'' are three different states (c). In each figure, all of the action-observation sequences q are identical to each other.

Definition 4. (*n-th-order ao-loop* ($n \geq 2$)) In environment E, if there exist n (≥ 2) different states $s^{(1)}, s^{(2)}, \dots$, and $s^{(n)}$ and one action-observation sequence q such that there is a transition path $s^{(1)} \to_q s^{(2)} \to_q \cdots \to_q s^{(n)} \to_q s^{(1)}$, then we call this path an ***n*-*th-order ao-loop*.**

For example, in Fig. 2(b), the state s transitions to s' with the action-observation sequence q, and the state s' transitions back to s' with the same action-observation sequence q. Since such a loop structure would make it difficult for the agent to distinguish the hidden states, we consider only environments that do not have ao-loops of any order in this paper.

Note that a 1st-order ao-loop (Fig. 2(a)) differs from higher-order ao-loops in the sense that a part of the transition path does not form a loop. For brevity, we still call it an ao-loop.

3.2. Goal of learning. In our approach, the goal of learning is to obtain a model that can perfectly predict the future. To be precise, we define the term *perfect prediction* as follows.

Definition 5. (*Perfect prediction*) We say that a model performs *perfect prediction* at time t if for any given future action sequence $(a(t), a(t+1), \dots, a(t+\tau-1))$ of any length τ (≥ 1) the model correctly predicts the future observation sequence $(o(t+1), o(t+2), \dots, o(t+\tau))$.

Given this definition, we call a model that performs perfect prediction a *perfect model*. The agent is therefore expected to acquire a perfect model through experience (i.e., via the history of actions and observations).

3.3. Agent. Using a bottom-up approach, the agent learns a hierarchical model of the environment while interacting with it.

Figure 3 shows an overview of the bottom-up learning approach. In what follows, superscripts (e.g., 1 in t_0^1) indicate the layer number. At time $t = 0$, the agent begins to interact with the environment. The initial time period from $t = 0$ to $t = t_0^1$ is when the *learning of layer 1* takes place. During this time period, the agent gathers

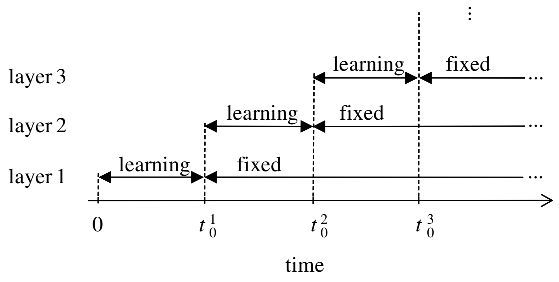


Fig. 3. Overview of the bottom-up learning approach. Each subsequent layer goes through a learning process and is then fixed such that layers above will complement the lower layers.

experience (i.e., actions and observations) and constructs layer 1, seeking better prediction performance. At time t_0^1, layer 1 is fixed.

The next time period from $t = t_0^1$ to $t = t_0^2$ is when the *learning of layer 2* takes place. During this period, the agent collects more experience and constructs layer 2, again seeking better prediction performance. Layer 2 is expected to play a complementary role in making predictions, i.e., it is expected to learn to predict what layer 1 cannot predict.

At time t_0^2, layer 2 is fixed, and the *learning of layer 3* begins. This process can be repeated to construct more layers (4, 5, and so forth), until perfect prediction is achieved.

In this paper, we consider the case in which each layer is a finite state machine (FSM), because perfect prediction can be performed by an FSM for any deterministic POMDP environment E. It is easy to prove this fact; we prove it in Section 3.3.2 after we define the lowest layer of the hierarchical model in Section 3.3.1.

3.3.1. Layer 1. Let the lowest layer, which we call layer 1, be an FSM defined as $M^1 := \langle B^1, A, O, U^1, P^1, b_0^1 \rangle$, where

- $B^1 := \{1, 2, \dots, |B^1|\}$ is a finite set of internal states, which we call *beliefs*,
- $A := \{1, 2, \dots, |A|\}$ is a finite set of actions,
- $O := \{1, 2, \dots, |O|\}$ is a finite set of observations,
- $U^1 := B^1 \times A \times O \to B^1$ is a belief update function,
- $P^1 := B^1 \times A \to O$ is a prediction function, and
- $b_0^1 \in B^1$ is the belief at layer 1's initial time t_0^1.

Here, A and O of M^1 are the same sets as those of environment E. The other entities of M^1 (i.e., B^1, U^1, P^1, and b_0^1) are to be set by some learning method (mentioned in Section 4.1).

The initial time of layer 1, t_0^1, is when layer 1 is fixed (Fig. 3). After this initial time, layer 1 behaves as shown in Fig. 4. At each time $t = t_0^1, t_0^1 + 1, t_0^1 + 2, \dots$, layer 1 receives action $a(t) \in A$ taken by the agent

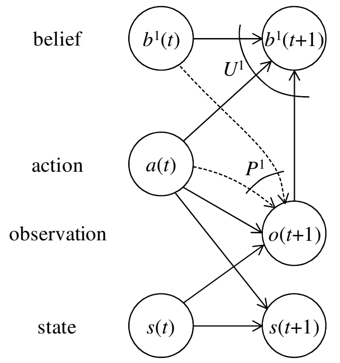


Fig. 4. Time slice of the POMDP environment with layer 1 of the agent.

and predicts the next observation $o(t+1) \in O$ to be $\tilde{o}(t+1) = P^1(b^1(t), a(t))$, where the tilde indicates a predicted value. After receiving the true observation $o(t+1)$ from the environment, layer 1 changes its belief $b^1(t)$ to $b^1(t+1) = U^1(b^1(t), a(t), o(t+1))$.

3.3.2. FSM can perform perfect prediction.

With M^1 defined, we prove that perfect prediction can be performed by an FSM as follows.

Theorem 1. *For any given environment E and any given time t (≥ 0), there exists an FSM M^1 that performs perfect prediction at time t.*

Proof. Let B^1, U^1, P^1, and b_0^1 of M^1 be such that M^1 simulates environment E; i.e., let B^1 be identical to the environment's state space S, let $U^1(s, a, o)$ be the same value as $\delta(s, a)$ for all $s \in S$, $a \in A$, and $o \in O$, and let $P^1(s, a)$ be the same value as $\gamma(s, a)$ for all $s \in S$ and $a \in A$. Let M^1's initial time t_0^1 be t. Let b_0^1 be the environment's state at t. Then, given any action sequence $(a(t), a(t+1), \ldots, a(t+\tau-1))$, the sequence of M^1's belief states $(b^1(t), b^1(t+1), \ldots, b^1(t+\tau-1))$ determined by U^1 is identical to that of the environment's states $(s(t), s(t+1), \ldots, s(t+\tau-1))$. Thus, future observations $(o(t+1), o(t+2), \ldots, o(t+\tau))$ are correctly predicted by P^1. ■

As is evident from the theorem above, perfect prediction can potentially be achieved by layer 1 alone. If perfect prediction is achieved by layer 1, then no higher layer is necessary. In practice, however, learning such a perfect model with a single layer is difficult. Typically, only some local structures would be learned by layer 1. Hence, we need higher layers to learn the rest. We define the higher layers in the next subsection.

3.3.3. Layer 2 and higher.

Let the l-th layer ($l \geq 2$) be an FSM defined as $M^l := \langle B^l, A^l, O, U^l, P^l, b_0^l \rangle$, where

- $B^l := \{1, 2, \ldots, |B^l|\}$ is a finite set of belief states,

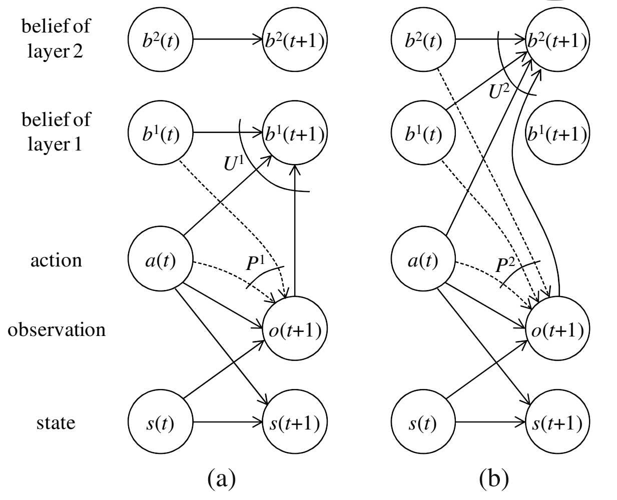


Fig. 5. Time slice of the POMDP environment with layers 1 and 2 of the agent. Usually, layer 2 does nothing; $o(t+1)$ is predicted by layer 1, and $b^2(t+1)$ equals $b^2(t)$ (a). When necessary, layer 2 gets involved in the prediction process; $o(t+1)$ is predicted by P^2, and $b^2(t+1)$ is determined by U^2 (b). Note that in (b) arrows going to $b^1(t+1)$ are omitted, but $b^1(t+1)$ is determined by U^1 in exactly the same manner as in (a).

- $A^l \subset B^1 \times B^2 \times \cdots \times B^{l-1} \times A$ is a finite set of actions,
- $O := \{1, 2, \ldots, |O|\}$ is a finite set of observations,
- $U^l := B^l \times A^l \times O \to B^l$ is a belief update function,
- $P^l := B^l \times A^l \to O$ is a prediction function, and
- $b_0^l \in B^l$ is the belief state at layer l's initial time t_0^l.

Here, O is the same set as that of environment E, and A^l is a set consisting of all possible configurations of the lower layers' beliefs $b^1 \in B^1, b^2 \in B^2, \ldots$, and $b^{l-1} \in B^{l-1}$ and action $a \in A$. A^l can be constructed by collecting all of the different values of vector $(b^1(t), b^2(t), \ldots, b^{l-1}(t), a(t))$ in the past history from $t = 0$ to the most recent value of t. Although the size of A^l may become exponentially large, usually it can be kept small; we discuss this issue in Section 5.2. The other entities of M^l (i.e., B^l, U^l, P^l, and b_0^l) are to be set by some learning method (described in Sections 4.2 and 4.3).

Layer l is fixed after some time period of learning, as shown in Fig. 3. Let t_0^l (≥ 0) be the time at which it is fixed. We call t_0^l the initial time of layer l. Layer l is fixed after layer $l-1$ is fixed; hence, we have $t_0^l \geq t_0^{l-1} \geq t_0^{l-2} \geq \cdots \geq t_0^1 \geq 0$.

We let layer l do some work only when it is necessary. This is because we want the higher layers to focus on predicting what the lower layers cannot predict (see Sections 4.2 and 4.3 for details). To describe how each layer works, consider, for example, the case of $l = 2$ (Fig. 5) where t is after t_0^2. In this time period, layer

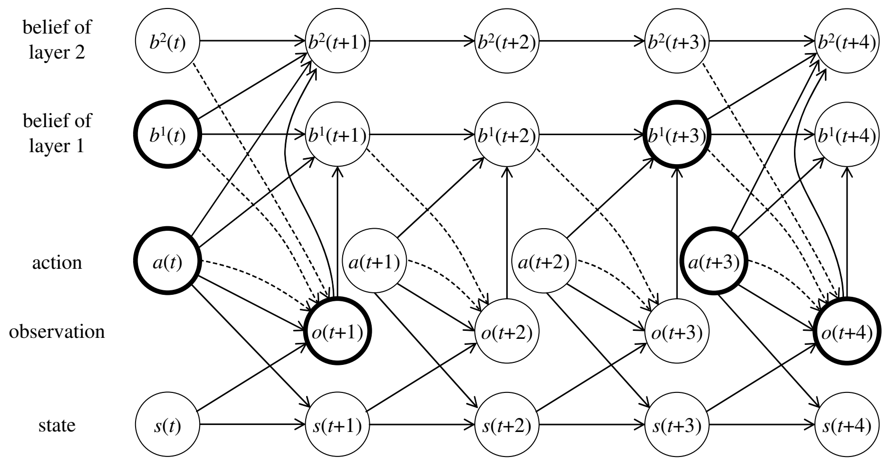


Fig. 6. Example of four time slices of the POMDP environment with layers 1 and 2 of the agent. In this example, layer 2 gets involved in the prediction process (as in Fig. 5(b)) at t and $t+3$, and does nothing (as in Fig. 5(a)) at $t+1$ and $t+2$. Each time slice in this figure is a replication of Fig. 5(a) or (b), but we include the arrows omitted in Fig. 5(b). Only the variables marked using thick circles directly affect layer 2's belief $b^2(\cdot)$.

1 has already been fixed and is working in the manner shown in Fig. 4; i.e., at each time t, layer 1 predicts the next observation $o(t+1)$ using P^1 and updates its belief $b^1(t)$ using U^1. At each time t, layer 2 chooses either to do nothing (Fig. 5(a)) or to get involved in the prediction process (Fig. 5(b)). Normally, layer 2 does nothing (Fig. 5(a)); i.e., the prediction of $o(t+1)$ made by layer 1 becomes the prediction of the whole model (i.e., layers 1 and 2), and the belief of layer 2 remains unchanged (i.e., $b^2(t) = b^2(t+1)$). When necessary, layer 2 gets involved in the prediction process (Fig. 5(b)). That is, layer 2 predicts $o(t+1)$ to be $\tilde{o}(t+1) = P^2(b^2(t), a^2(t))$, where $a^2(t) := (b^1(t), a(t))$. After receiving the true observation $o(t+1)$ from the environment, the belief of layer 2 is updated to $b^2(t+1) = U^2(b^2(t), a^2(t), o(t+1))$.

Next, we restate the above for general l (≥ 2). Normally, layer l does nothing, and the prediction of $o(t+1)$ made by layers $1, 2, \ldots,$ and $l-1$ becomes that of the whole model (i.e., layers $1, 2, \ldots,$ and l). The belief of layer l remains unchanged; i.e., $b^l(t) = b^l(t+1)$. When necessary, layer l predicts $o(t+1)$ to be $\tilde{o}(t+1) = P^l(b^l(t), a^l(t))$, where $a^l(t) := (b^1(t), b^2(t), \ldots, b^{l-1}(t), a(t))$. After receiving the true observation $o(t+1)$, the belief of layer l is updated to $b^l(t+1) = U^l(b^l(t), a^l(t), o(t+1))$.

At each time t, the agent determines whether layer l should get involved in the prediction process; this determination is based on the values of $b^1(t), b^2(t), \ldots, b^{l-1}(t)$, and $a(t)$. Further details are described in the next section.

We explain the entire process using the example in Fig. 6. In this figure, there are two layers and four time slices from t to $t+4$. Suppose that, at time t, the values of $b^1(t)$, $b^2(t)$, and $s(t)$ have already been fixed. The agent selects action $a(t)$ in a certain way (which we do not consider in this paper) and determines whether layer 2 should get involved, based on the values of $b^1(t)$ and $a(t)$. In the example in Fig. 6, layer 2 gets involved. Consequently, layer 2 predicts the next observation $o(t+1)$, and after the true observation $o(t+1)$ is given from the environment, the beliefs of layers 1 and 2 are updated to $b^1(t+1)$ and $b^2(t+1)$, respectively. At the same time, the state of the environment is updated to $s(t+1)$. At time $t+1$, the agent selects the next action $a(t+1)$ and determines whether layer 2 should get involved, based on the values of $b^1(t+1)$ and $a(t+1)$. In the example in Fig. 6, layer 2 does not get involved, i.e., it does nothing. The next observation $o(t+2)$ is predicted by layer 1, and the belief of layer 2 remains unchanged (i.e., $b^2(t+2) = b^2(t+1)$). The belief of layer 1 and the state of the environment are updated to $b^1(t+2)$ and $s(t+2)$, respectively. The process continues in this manner. In the example in Fig. 6, layer 2 does nothing at $t+2$ and gets involved at $t+3$.

Our model is related to POSMDPs (White, 1976; Mahadevan, 1998; Lim *et al.*, 2011). Note that in the example in Fig. 6 only the variables shown in thick circles, i.e., $b^1(\cdot)$, $a(\cdot)$, and $o(\cdot)$, directly affect layer 2's belief $b^2(\cdot)$. These variables ($b^1(\cdot)$, $a(\cdot)$, and $o(\cdot)$) can be considered to obey a jump process in which, given an action $a(t)$ at time t, layer 1's belief $b^1(t)$ emits $o(t+1)$ and changes sometime later to $b^1(t+\tau)$, where $\tau = 3$ in this example. This process is not exactly a POSMDP, but if we include the hidden state variables $s(t)$ and $s(t+\tau)$ and the actions $a(t+1)$, $a(t+2)$, $\cdots$, and $a(t+\tau-1)$, then we obtain a POSMDP in which, given $a(t)$, $a(t+1)$, $\cdots$, and $a(t+\tau-1)$, the state of the process $(b^1(t), s(t))$ changes to $(b^1(t+\tau), s(t+\tau))$ emitting $o(t+1)$.

4. Bottom-up learning algorithm

Our bottom-up learning algorithm is shown in Algorithm 1; each step of this algorithm is described below.

4.1. Making M^1. The first step of our learning algorithm (Step 1 in Algorithm 1) is the process of learning at the lowest layer M^1. Although this is an important first step of the entire learning process, we do not focus on it in this paper. Any method can be used to learn M^1. For example, the Baum–Welch algorithm can be used to learn an input–output HMM, which, after learning, is converted into an FSM by discretizing the belief space. Alternatively, the FSM's parameter values can be directly searched by gradient ascent methods, genetic algorithms, or other such methods.

4.2. Adding M^2 on top of M^1. Suppose that M^1 has already been learned and fixed by some method and that perfect prediction has not been achieved by M^1. In this section, we consider how to add a second layer M^2 on top of M^1.

4.2.1. Overview. Before describing the details, we provide an overview by using an example. Consider a mobile agent that is learning a map of a building's layout shown in Fig. 7(a). This is a grid map, and each cell (□) is a state hidden from the agent. We suppose that the cells with the numbers 1–5 represent floors and those with the number 6 or 7 represent staircases.

Suppose that this map is learned only partially by M^1. For example, let us consider an M^1 whose belief update function U^1 is such that belief $b^1(t)$ becomes identical to $o(t)$ when $o(t) \in \{1, 2, \dots, 7\}$ and to $b^1(t-1)$ when $o(t)$ is "no move." If we use this M^1, then $b^1(t)$ is always identical to the number that is observed in the cell where the agent is located, as shown in Fig. 7(b). This $b^1(t)$ represents the current position in each corridor or staircase; hence, the model successfully predicts the future observations as long as the agent moves inside a corridor or staircase; however, since it does not represent the exact position within the entire map, the model cannot always make a correct prediction.

When adding M^2 on top of M^1, it would be desirable if we could ignore the things that have already been learned by M^1 and allow M^2 to focus on learning what has not been learned. In the example illustrated in Fig. 7(b), since M^1 has already learned the map of each corridor and staircase, it would be desirable if M^2 could focus on learning how the corridors and staircases are connected to each other.

To accomplish this, we must first identify what can be safely ignored. The idea behind our algorithm is to ignore the time periods during which *full observability* is recovered by M^1. In the example shown in Fig. 7(b), during the time periods when the agent is moving inside a corridor or staircase, full observability is recovered by M^1 in the sense that at each time t, belief $b^1(t)$ has sufficient information for the agent to determine the position within the corridor (e.g., if $b^1(t) = 1$ holds, the agent knows that it is located in the leftmost position of a corridor on some floor). This suggests that during these time periods, there is nothing left for M^2 to learn. We will prove, in Theorems 2 and 3 introduced in the next section, that these time periods can be safely ignored when M^2 is learning. It is not necessary for M^2 to do anything during these time periods, and M^2 can focus on predicting the observations at other time steps. Perfect prediction is achieved once the belief of M^2 becomes capable of distinguishing corridors and staircases, as illustrated in Fig. 7(c).

Algorithm 1. Bottom-up learning.

```
Step 1. Learn $M^1$ by any method.
Step 2. Explore the environment and let the recent history $h$ be $h = (b^1ao(t-T+1), b^1ao(t-T+2), \dots, b^1ao(t))$.
Step 3. Find OP transitions by counting $N(b^1, a, o)$.
Step 4. Find OPM transitions.
Step 5. Obtain $h' = (b^1ao(t_1), b^1ao(t_2), \dots, b^1ao(t_n))$ from $h$ by discarding all $b^1ao(\cdot)$ associated with the OPM transitions.
Step 6. Optimize $M^2$ using $h'$.
Step 7. To construct higher layer $M^l$ ($l \geq 3$), repeat Steps 2–6 with $b^1$ and $M^2$ replaced with $b^{1:l-1}$ and $M^l$, respectively.
```

In the next section, we present a method for determining the time periods during which full observability is recovered.

4.2.2. Discarding ignorable parts. As described above, we wish to ignore time periods during which full observability is recovered. One important clue for judging whether full observability is recovered is the *predictability* of the observation at the next time step. To be precise, before we define the notion of predictability (i.e., Definition 7), we introduce some fundamental notions regarding the combined system of environment E and layer 1.

Definition 6. (*Combined POMDP and its extended state*) For any environment E and any FSM M^1, we call a POMDP consisting of E and M^1 a *combined POMDP*. The state of the combined POMDP is $(s, b^1) \in S \times B^1 := \{(s, b^1) \mid s \in S, b^1 \in B^1\}$, which we call an *extended state*. The action and observation spaces of the combined POMDP are the same as those of environment E. The extended state (s, b^1) is changed to $(s', b^{1\prime})$ by action $a \in A$ of the agent, where s' is the next state of the environment (i.e., $s' = \delta(s, a)$), and $b^{1\prime}$ is the next belief

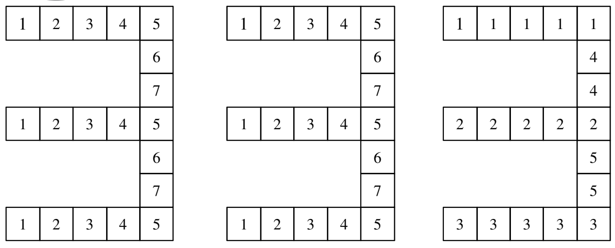


Fig. 7. Mapping a three-story building environment. Map of the environment; in each cell (□), the agent can move left, right, up, or down; the number written in each cell is the observation given to the agent when the agent enters the cell; if the agent attempts to take an impossible action (e.g., going up in a cell with the number 1), a different value indicating "no move" is observed (a). Example of M^1's belief, in which the number written in each cell is the value of $b^1(t)$ when the agent is located in the cell (b). Example of M^2's belief, in which the number written in each cell is the value of $b^2(t)$ when the agent is located in the cell (c).

of layer 1 (i.e., $b^{1\prime} = U^1(b^1, a, o)$, where $o = \gamma(s, a)$). The observation emitted by the combined POMDP is the same as that of environment E. The initial extended state is $(s(t_0^1), b_0^1)$, where $s(t_0^1)$ is the environment's state at the initial time of layer 1.

Further, we say that state s *co-occurs* with belief b^1 when the extended state is (s, b^1). Given these definitions, we define predictability below.

Definition 7. (*Observation-predictable (OP) transition*) In any strongly connected region of the space of extended states $S \times B^1$, if selecting action a in belief b^1 always results in an identical observation regardless of which state s is co-occurring with belief b^1, then we say that the pair (b^1, a) causes an *observation-predictable transition*, or *OP transition* for short. We write this transition as $(b^1, a) \to_{\text{OP}} b^{1\prime}$, where $b^{1\prime}$ is the next belief after b^1 and a.

Note that in Definition 7 we consider only strongly connected regions of $S \times B^1$. This is because the other (i.e., transient) regions are difficult to learn since the agent potentially moves through these regions only once.

Whether a transition is OP or not can be judged from the history that the agent has experienced while interacting with the environment. This corresponds to Steps 2 and 3 of Algorithm 1. In Step 2, suppose that the agent has explored the environment and recorded the history of M^1's beliefs, actions, and observations up to time t. Let the belief-action-observation history of recent T time steps be

$$h := (b^1ao(t-T+1), b^1ao(t-T+2), \ldots, b^1ao(t)), \quad (1)$$

where $b^1ao(\cdot)$ is the abbreviated notation for a tuple $(b^1(\cdot), a(\cdot), o(\cdot + 1))$. Since OP transition is a notion regarding a strongly connected region of $S \times B^1$, we must discard initial parts of the experienced history to exclude the time period before the agent falls into a strongly connected region. There are multiple ways to do this, but in the numerical experiments in Section 5 below we simply discarded the initial 1/10 of the history; i.e., we set $T := 9t/10$ in Eqn. (1). By doing this, if t is sufficiently large, history h includes only the experience after the agent falls into a strongly connected region.

In Step 3, for every pair of belief $b^1 \in B^1$ and action $a \in A$, let $N(b^1, a, o)$ for each $o \in O$ be the number of occasions in which o is observed immediately after b^1 and a. We use history h to count $N(b^1, a, o)$. If $N(b^1, a, o) > 0$ for only a single observation o and $N(b^1, a, o) = 0$ for other o's, then (b^1, a) is identified as the pair that causes an OP transition. Although misidentifications can occur if history h is too short, correct identifications will be made after h becomes sufficiently long.

The notion of OP transition introduced above is an important clue, but it is not sufficient for us to know whether full observability is recovered. We need additional notions regarding mutuality, which we describe below.

Definition 8. (*Mutually observation-predictably transitionable*) For any two beliefs b^1 and $b^{1\prime} \in B^1$, we say that b^1 and $b^{1\prime}$ are *mutually observation-predictably transitionable* if we have both $(b^1, \exists a) \to_{\text{OP}} b^{1\prime}$ and $(b^{1\prime}, \exists a') \to_{\text{OP}} b^1$.

Definition 9. (*OPM transition*) For any two beliefs b^1 and $b^{1\prime} \in B^1$ and action $a \in A$, if (1) (b^1, a) causes an OP transition to $b^{1\prime}$ and (2) b^1 and $b^{1\prime}$ are mutually observation-predictably transitionable, then we say that (b^1, a) causes an *OPM transition* to $b^{1\prime}$. We write this transition as $(b^1, a) \to_{\text{OPM}} b^{1\prime}$. If (b^1, a) causes an OPM transition and o is the observation that is obtained immediately after (b^1, a), then we say that (b^1, a, o) is associated with the OPM transition.

Note that in Definition 9 the OPM acronym combines the observation-predictable transition and the mutually observation-predictably transitionable beliefs.

Also note that, from Definitions 8 and 9, OPM transitions can be found using OP transitions. This corresponds to Step 4 of Algorithm 1.

With these notions defined, we can identify the time periods during which full observability is recovered by M^1. In history h in Eqn. (1), some elements $b^1ao(\cdot)$ would be associated with OPM transitions, while others would not. The theorem below states that full observability is recovered by M^1 during the time period in which every $b^1ao(\cdot)$ is associated with an OPM transition.

Theorem 2. *Consider a combined POMDP consisting of any environment E without ao-loops and any FSM M^1. After the agent falls into a strongly connected region of $S \times B^1$ of the combined POMDP, if there is any time period $t_1 \leq t \leq t_2$ such that every $b^1ao(t)$, $t \in [t_1, t_2]$, is associated with an OPM transition, then for any t_i and $t_j \in [t_1, t_2+1]$ we have $s(t_i) \neq s(t_j) \Rightarrow b^1(t_i) \neq b^1(t_j)$.*

Proof. See Appendix. ■

The main idea of the proof is as follows. First, the space of extended states can be partitioned into blocks, which we call *x-blocks*; within each block, all of the extended states are connected by undirected OPM transitions. Next, we can prove that for any two extended states (s, b^1) and $(s', b^{1\prime})$ within an x-block we have $s \neq s' \Rightarrow b^1 \neq b^{1\prime}$. Additionally, during any time period $t_1 \leq t \leq t_2$ such that every $b^1ao(t)$, $t \in [t_1, t_2]$, is associated with an OPM transition, the extended state $(s(t), b^1(t))$ stays within a single x-block. We can use these two facts to prove Theorem 2.

Note that the final part of Theorem 2 (i.e., $s(t_i) \neq s(t_j) \Rightarrow b^1(t_i) \neq b^1(t_j)$) means full observability, since it states that there is no such case in which different states are confusingly represented by a single belief.

To learn M^2, we ignore the fully observable periods and use only the remaining parts. This corresponds to Step 5 in Algorithm 1. Within history h in Eqn. (1), let t_i ($i = 1, 2, \ldots, n$) be the time steps in each of which $b^1ao(t_i)$ is not associated with an OPM transition, where $t - T + 1 \leq t_1 < t_2 < \cdots < t_n \leq t$. We ignore the other time steps and use

$$h' = (b^1ao(t_1), b^1ao(t_2), \ldots, b^1ao(t_n)) \qquad (2)$$

for the learning process of M^2. In the next section, we prove that this h' is sufficient for the agent to use to successfully learn M^2.

4.2.3. Optimization of M^2.

The next step of our learning algorithm, i.e., Step 6 in Algorithm 1, is to optimize M^2. The optimization is performed such that M^2 can predict, in collaboration with M^1, future observations as correctly as possible.

Before describing the optimization process itself, we first state how M^2 collaborates with M^1. As mentioned in Section 3.3.3, layer 2 either does nothing (Fig. 5(a)) or gets involved in the prediction (Fig. 5(b)) at each time. We let layer 2 do nothing when $(b^1(t), a(t))$ causes an OPM transition, whereas we let layer 2 get involved in the prediction when $(b^1(t), a(t))$ does not cause an OPM transition. More specifically, layers 1 and 2 work in collaboration, as shown in Algorithm 2.

In what follows, we assume that, whenever $(b^1(t), a(t))$ causes an OPM transition, M^1 correctly predicts $o(t+1)$. We need this assumption because when $(b^1(t), a(t))$ causes an OPM transition, we adopt the prediction made by M^1 as the prediction of the whole model (Step (2a) in Algorithm 2). It is easy for M^1 to satisfy this assumption, since, by the definition of OPM transitions, $o(t+1)$ is uniquely determined when $(b^1(t), a(t))$ causes an OPM transition.

Algorithm 2. Making predictions via layers 1 and 2 after $t = t_0^2$ (see Section 4.3 for layers higher than layer 2).

```
Do the following at each time $t = t_0^2, t_0^2+1, t_0^2+2, \ldots$.
Step 1. Layer 1 makes a prediction of the next observation $o(t+1)$ by $P^1$. After obtaining the true observation, layer 1's belief $b^1(t)$ is updated to $b^1(t+1)$ by $U^1$.
Step 2. Layer 2 does either (2a) or (2b), depending on the value of $(b^1(t), a(t))$.
(2a) If $(b^1(t), a(t))$ causes an OPM transition, then the prediction of $o(t+1)$ made by layer 1 is used as the prediction of the whole model; $M^2$ does nothing, and its belief $b^2(t)$ remains unchanged (i.e., $b^2(t+1) = b^2(t)$).
(2b) If $(b^1(t), a(t))$ does not cause an OPM transition, then the prediction of $o(t+1)$ made by $M^1$ is discarded, and the prediction made by layer 2, i.e., $P^2(b^2(t), a^2(t))$, is used instead, where $a^2(t) := (b^1(t), a(t))$. Belief $b^2(t)$ is updated to $b^2(t+1) = U^2(b^2(t), a^2(t), o(t+1))$ after obtaining the true observation $o(t+1)$.
```

Using Algorithm 2, perfect prediction can be attained by M^1 and M^2, as stated in the following theorem.

Theorem 3. *Consider any environment E without ao-loops and any FSM M^1 that correctly predicts $o(t+1)$ whenever $(b^1(t), a(t))$ causes an OPM transition. For any time t after the agent falls into a strongly connected region of $S \times B^1$ of the combined POMDP consisting of E and M^1, there exists an $M^2 = \langle B^2, A^2, O, U^2, P^2, b_0^2 \rangle$ that performs perfect prediction at time t, collaborating with M^1 in the manner described in Algorithm 2.*

Proof. See Appendix. ■

In the proof provided in Appendix, we prove the existence of an M^2 that performs perfect prediction by constructing an example.

The above theorem tells us that there exists an M^2 that performs perfect prediction, but it does not state how to learn such an M^2. There can be multiple ways to learn it. In the experiments in Section 5 below, we used the following method. First, recall that, in Algorithm 2, M^2 must predict the next observation $o(t+1)$ when and only when $(b^1(t), a(t))$ does not cause an OPM transition. Thus, we used the history of non-OPM transitions, i.e., $h' = (b^1ao(t_1), b^1ao(t_2), \ldots,$

$b^1ao(t_n))$ of Eqn. (2), as the training data for M^2. At each time t_i ($i = 1, 2, \dots, n$), we want M^2 to correctly predict the next observation $o(t_i + 1)$ given current belief $b^1(t_i)$, action $a(t_i)$, and previous history $(b^1ao(t_1), b^1ao(t_2), \dots, b^1ao(t_{i-1}))$. We optimized the parameters of M^2 such that the prediction is correct in as many time steps as possible. For parameter optimization, we used a simple gradient ascent method, which is described in Section 5.2.

4.3. Constructing M^3 and higher layers. Constructing an M^2 that performs perfect prediction may still be difficult to accomplish. If necessary, we can add higher layers M^3, M^4, and so on. This corresponds to Step 7 of Algorithm 1.

Let us begin by considering the case of adding M^3 on top of M^1 and M^2. Instead of two FSMs M^1 and M^2, we can consider an equivalent single FSM, which we denote as $M^{1:2}$. In $M^{1:2}$, the belief at time t is $b^{1:2}(t) := (b^1(t), b^2(t))$, the space of the belief states is $B^1 \times B^2$, the action space is A, the observation space is O, the belief update and prediction functions are defined according to Algorithm 2, and the initial belief is $b^{1:2}(t_0^2) := (b^1(t_0^2), b^2(t_0^2))$.

Adding M^3 on top of layers M^1 and M^2 is equivalent to adding M^3 on top of the single layer $M^{1:2}$. Thus, all of the definitions, theorems, and Algorithm 2 of Section 4.2 above are valid if we replace M^1 with $M^{1:2}$. Therefore, Steps 2–6 of Algorithm 1 can be repeated in exactly the same manner, except that b^1 and M^2 are replaced with $b^{1:2}$ and M^3, respectively.

In general, adding M^l on top of layers $M^1, M^2, \dots,$ and M^{l-1} is equivalent to adding M^l on top of a single layer $M^{1:l-1}$ that behaves in exactly the same way as layers $M^1, M^2, \dots,$ and M^{l-1}. Given this, Steps 2–6 of Algorithm 1 can again be repeated in exactly the same way, except that b^1 and M^2 are replaced with $b^{1:l-1}$ and M^l, respectively. This is Step 7 in Algorithm 1.

As stated in the theorem below, we note that if $b^{1:l-1}ao(t)$ is associated with an OPM transition, then $b^{1:l}ao(t)$ is associated with an OPM transition. Therefore, when we create layer l, we do not need all time steps of the recent history $h = (b^{1:l-1}ao(t-T+1), b^{1:l-1}ao(t-T+2), \dots, b^{1:l-1}ao(t-1), b^{1:l-1}ao(t))$ in Step 2 of Algorithm 1; time steps that have been judged by lower layers $1, 2, \dots,$ and $l-2$ to be associated with an OPM transition can be safely excluded from h. Thus, the history h for making M^l can become much shorter, reducing the computational time of subsequent Steps 3–5.

Theorem 4. *For any environment E and any $l \geq 2$, suppose we have FSMs $M^1, M^2, \dots,$ and M^l that are working as in Algorithm 2. For any time $t \geq t_0^l$, if $b^{1:l-1}ao(t)$ is associated with an OPM transition, then $b^{1:l}ao(t)$ is associated with an OPM transition.*

Proof. See Appendix. ■

The main idea of the proof is as follows. We begin with the case where $l = 2$. First, we have a bidirectionality of OPM transitions. That is, for any $b^1 \in B^1$ and $a \in A$, if (b^1, a) causes an OPM transition to $b^{1\prime}$, then there exists an action $a' \in A$ such that $(b^{1\prime}, a)$ causes an OPM transition returning to b^1. Next, $b^2(t)$ does not change with an OPM transition. From these two facts, we can prove Theorem 4 for the case of $l = 2$. For $l > 2$, we can prove the theorem by replacing $M^1, M^2, \dots,$ and M^{l-1} in the theorem with an equivalent $M^{1:l-1}$.

4.4. Time complexity. The time complexity of each step of Algorithm 1 is as follows.

Step 1 of Algorithm 1 can be accomplished by any method, and the time complexity depends on the technique used. Step 2 requires $O(|h|)$ time to store the training data h. Step 3 requires $O(|h|)$ time to count $N(b^1, a, o)$ from h and $O(|B^1||A||O|)$ time to determine whether each (b^1, a) causes an OP transition.

There are multiple ways to find the OPM transitions in Step 4, but the following is one of the easiest. Let us consider a $|B^1| \times |B^1|$ binary matrix X. First, we set all the elements of X to 0. Next, for every OP transition $(b^1, a) \to b^{1\prime}$ found in Step 3, we set the $(b^1, b^{1\prime})$-th element of X to 1. Note that if two beliefs b^1 and $b^{1\prime} \in B^1$ are mutually observation-predictably transitionable (Definition 8), then both the $(b^1, b^{1\prime})$-th and $(b^{1\prime}, b^1)$-th elements of X are set to 1. Using this matrix X, each OP transition $(b^1, a) \to b^{1\prime}$ is judged to be an OPM transition if and only if the $(b^{1\prime}, b^1)$-th element of X is 1. Since there are at most $|B^1||A|$ OP transitions, if the memory access to each element of X can be done in a constant time regardless of $|B^1|$, then we require $O(|B^1||A|)$ time for Step 4.

Step 5 requires $O(|h|)$ time because we must determine whether each element of h is associated with the OPM transitions.

Step 6 can be implemented using different learning methods, each of which may have a different time complexity. Most methods require at least $O(|h'|)$ time to process the training data h' (for example, our simple greedy search method used in Section 5 requires $O(|h'| + |B^2||A^2||O|)$ time for each iteration). Thus, it is important that we keep $|h'|$ small by ignoring the unimportant parts of h.

To construct higher layers M^l ($l \geq 3$) in Step 7, we must repeat Steps 2–6. For layer l, the time complexities of Steps 2–6 are $O(|h|)$, $O(|h| + |B^l||A^l||O|)$, $O(|B^l||A^l|)$, $O(|h|)$, and (at least) $O(|h'|)$, respectively. Note that Theorem 4 implies that the length of h for higher layers can be made much shorter than for layer 1.

5. Numerical experiments

5.1. Examples of environments without ao-loops. In this section, we describe two classes of environments that have no ao-loops, which we use in the numerical experiments described in the next section. Note that these classes are just two examples; there are other classes of environments without ao-loops.

Example 1. (*Two-dimensional grid mazes with special observations for impossible moves*) Let C_1 be a class of environments described as follows. Environment $E \in C_1$ is a two-dimensional grid maze environment (e.g., as shown in Fig. 7(a)). In the maze, the agent can move via four actions, right, left, up, and down, each of which brings the agent to an adjacent grid cell. The set of observations O is divided into two disjoint sets O_{stay} and O_{move}. When the agent takes an action, if the motion intended by the action is impossible (e.g., moving up from a cell labeled 1 in Fig. 7(a) is impossible), the agent stays in the current cell and obtains some observation $o \in O_{\text{stay}}$. If the motion intended by the action is possible, the agent moves to the new cell and obtains some observation $o \in O_{\text{move}}$.

Theorem 5. *Any environment in C_1 does not have an ao-loop.*

Proof. Consider any environment in C_1 and any action-observation sequence $q = (a_1, o_1, a_2, o_2, \ldots, a_n, o_n)$ that occurs in the environment. For each action a_i $(i = 1, 2, \ldots, n)$ of q, if subsequent observation o_i is in O_{move}, then we know that the agent's position is changed as intended. Conversely, if the subsequent observation o_i is in O_{stay}, then we know that the agent's position is unchanged. Thus, the displacement $(\Delta x, \Delta y)$ of the agent's position in the maze caused during sequence q is uniquely determined, regardless of the state in which the agent is located before q.

Therefore, in any environment in C_1, if there are two different states s and $s' \in S$ and sequence q such that $s \to_q s'$, then we know that $(\Delta x, \Delta y) \neq (0, 0)$ for q, which implies that $s' \to_q s'$ is impossible. Thus, there is no such path $s \to_q s' \to_q s'$; i.e., there is no 1st-order ao-loop. Similarly, we can prove that there are no 2nd- or higher-order ao-loops. ■

Example 2. (*Two-dimensional grid mazes of Example 1, with jumps*) In grid mazes, let us call a movement to a non-adjacent cell a *jump*. Let C_2 be another class of environments described as follows. Environment $E \in C_2$ (e.g., as shown in Fig. 8) is the same maze environment as Example 1, except that the agent can move to non-adjacent cells under the condition that if taking action a in state s causes a jump, then the resultant action-observation pair (a, o), where $o = \gamma(s, a)$, should be unique to (s, a), that is, no other state-action pair should result in (a, o).

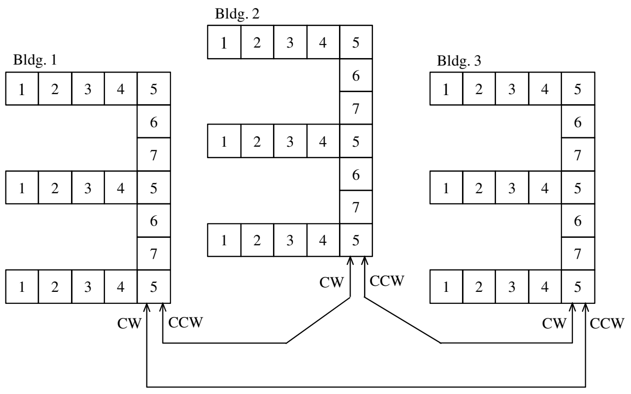


Fig. 8. Environment consisting of three buildings. Each building is the same as that of Fig. 7(a). The buildings are arranged in a circle. In addition to the four actions, right, left, up, and down, there are two actions, which we call clockwise (CW) and counterclockwise (CCW) jumps, each of which causes the agent to move to the neighboring building in the CW and CCW directions, respectively. These CW and CCW actions are valid only when the agent is in the bottom-right cell (which represents the entrance) of each building. After a valid CW or CCW action, the agent observes the number (i.e., #1, #2, or #3) of the building to which the agent moved. In all the other cells, CW and CCW actions are invalid; i.e., the agent stays in the same cell and observes “no move.”

Theorem 6. *Any environment in C_2 does not have an ao-loop.*

Proof. Consider any environment in C_2 and any action-observation sequence $q = (a_1, o_1, a_2, o_2, \ldots, a_n, o_n)$ that occurs in the environment. Because of the condition imposed on the environments in C_2, we know when and from where jumps occur in q. For example, if (a_1, o_1) of q is an action-observation pair that is unique to a jump caused by taking action a_1 in some state s, then we know that a jump occurs in the first time step of q and that the jump is from the state s.

Let i be the first time step at which a jump occurs in q (if no jump occurs in q, then the proof of Theorem 5 applies). As mentioned above, we know that the state s from which the jump occurs at step i is uniquely determined. In addition, since state transitions before step i are ordinary (i.e., non-jump) transitions, we know from the proof of Theorem 5 that the displacement $(\Delta x, \Delta y)$ caused by these transitions is uniquely determined. From these two facts, we know that the state from which sequence q starts is uniquely determined.

Therefore, in any environment in C_2, there is no such path $s \to_q s' \to_q s'$, because in this path q starts from different states s and s'. Thus, there is no 1st-order ao-loop. Similarly, we can prove that there is no 2nd- or higher-order ao-loops. ■

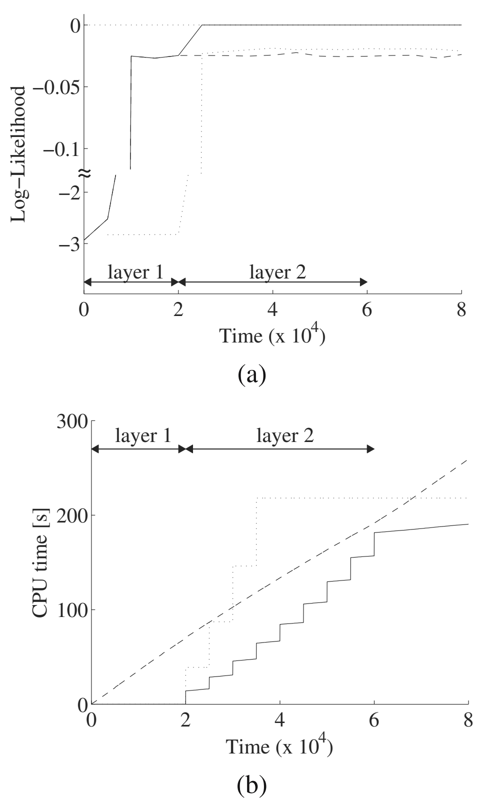


Fig. 9. Results of learning a building environment. Each line shows the average of 20 runs with different random seeds. Log-likelihood of the prediction of the next observation $o(t+1)$; the solid line is the result of using layers 1 and 2, whereas the broken line is the result of using only layer 1. Layer 1 learned from $t=0$ to 20,000, and layer 2 learned from $t=20{,}000$ to 60,000. The optimization of U^2 and P^2 was performed every 5,000 time steps. The dotted line is the result of the DBN-based learning method (Theocharous *et al*., 2004) (a). Cumulative CPU time spent in each layer; the broken and solid lines represent the CPU time spent in layers 1 and 2, respectively. The dotted line is the result of the DBN-based learning method, in which the actual CPU time is 1,500 times longer than that indicated by the dotted line (b).

5.2. Experiments and results. We tested our bottom-up learning algorithm in two environments.

5.2.1. Four-story building. One environment is the same as that of Fig. 7(a), except that there are 4 floors instead of 3 and that in each floor there are 10 cells instead of 5. The observations obtained in the staircase cells are 11 and 12 instead of 6 and 7. Thus, we have $|S|=46, |A|=4$, and $|O|=13$ in this environment (here, we have $|O|=13$ because there are 12 observations obtained after the agent moves to a different cell and 1 "no move" observation). This environment belongs to class C_1 in Section 5.1; however, the agent is not aware of this fact. Further, the agent is not given any prior knowledge regarding which observations are in O_{stay} or O_{move}.

We used two layers in this case. For layer 1, we prepared B^1, U^1, and b_0^1 such that the observation obtained after a successful movement directly becomes the belief (i.e., as shown in Fig. 7(b)). These are prepared by hand. Layer 1 learned P^1 only.

For layer 2, we used our learning algorithm. We set $b_0^2 = 1$ without loss of generality and set $|B^2| = 4$. The recent history h in Eqn. (1) was prepared by discarding the initial 1/10 of the entire history from $t=0$ to the current time. For the learning of U^2 and P^2 (i.e., Step 6 of Algorithm 1), we used a simple greedy search method. More specifically, we repeated the following steps until convergence was achieved: (1) modify U^2 by setting a different value to a parameter of U^2, (2) learn P^2 using the modified U^2, and (3) accept (or reject) the modified U^2 if the rate of correct prediction improved (or did not improve). The parameter to be modified was selected in turn. In each experiment, this greedy search was carried out every 5,000 time steps. For each time the search was carried out, the same search was repeated 10 times with different initial parameter values in order to (hopefully) escape local optima.

The results are shown in Fig. 9. The perfect prediction was achieved by $t=25{,}000$ (Fig. 9(a)). By $t=25{,}000$, approximately 90 seconds of CPU time were spent in layer 1 and 30 seconds in layer 2 (Fig. 9(b)). The CPU time includes both the time spent for learning and the time spent for prediction. We used an Intel Core i7 3.4 GHz PC. The program was written in MATLAB.

As shown in Theorems 2 and 3, unimportant parts of the training data can be safely ignored when layer 2 is learning. For example, at time $t=60,000$ in this experiment (which is the final time step of layer 2's learning period), the length of h in Step 2 of Algorithm 1 was $54,000$, because we discarded the initial 1/10 of the entire history (see Section 4.2.2 for details). In contrast, the length of h' in Step 5 of Algorithm 1 was $4,080 \pm 275$ (mean $\pm$ standard deviation of 20 runs). Thus, $92.4 \pm 0.5\%$ of the training data were safely ignored.

For comparison, we include results of the well-known DBN-based method (Theocharous *et al*., 2004) in Fig. 9. We used the MATLAB code in the BNT toolbox (Murphy, 2002). Learning was carried out every 5,000 time steps after $t=20{,}000$; for each time the learning was performed, the same learning process was repeated 10 times with different initial values, in order to (hopefully) escape local optima. Prediction performance was slightly better than that of layer 1 (Fig. 9(a)); perfect prediction was not achieved in all of the 20 runs that we conducted. The log-likelihood at the final time step (i.e., $t=80{,}000$) was statistically significantly lower than that of layer 2 ($p<0.001$, t-test) and marginally higher than that of layer 1 ($p=0.06$, t-test). Further, much

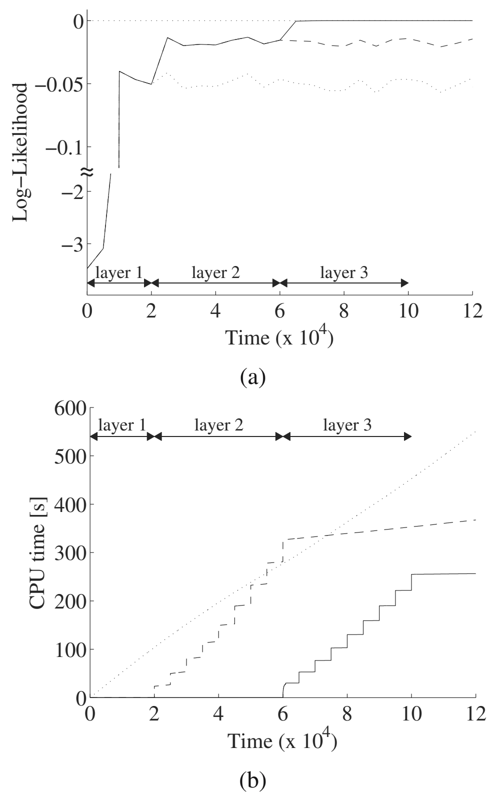


Fig. 10. Results of learning a 10-building environment: each line is the average of 20 runs conducted with different random seeds. Log-likelihood of the prediction; the solid line is the result of using all layers 1, 2, and 3, the broken line is the result of using only layers 1 and 2, and the dotted line is the result of using only layer 1. Layer 1 learned from $t = 0$ to 20,000, layer 2 learned from $t = 20{,}000$ to 60,000, and layer 3 learned from $t = 60{,}000$ to 100,000 (a). Cumulative CPU time spent in each layer; the dotted, broken, and solid lines represent the CPU time spent in layers 1, 2, and 3, respectively (b).

CPU time was spent—more than 300,000 seconds by $t = 35{,}000$; therefore, we stopped the learning process at $t = 35{,}000$. Thus, our method outperformed the DBN-based approach, at least in this environment. To be fair, we should note that the DBN-based method is applicable for learning general stochastic POMDPs. We also note that there is a possibility that the method can be sped up, as noted by Wakabayashi and Miura (2012).

5.2.2. Ten four-story buildings.

The second environment is the same multi-building environment as shown in Fig. 8, except that each building is a four-story building from Section 5.2.1 above and that there are 10 buildings. Thus, we have $|S| = 460, |A| = 6$, and $|O| = 23$ in this environment (here, we have $|O| = 23$ because there are 12 within-building observations, 10 building-number observations, and 1 "no move" observation). This environment belongs to class C_2 in Section 5.1, although this fact is not given to the agent.

We used three layers in this case. Layers 1 and 2 are exactly the same as those described in Section 5.2.1. For layer 3, we set $b_0^3 = 1$ without loss of generality, and we set $|B^3| = 10$. For the learning of U^3 and P^3 (i.e., Step 6 in Algorithm 1), we used the same greedy method as we did for layer 2.

The results are shown in Fig. 10. Perfect prediction was achieved by $t = 70{,}000$ (Fig. 10(a)). By $t = 70{,}000$, approximately 330 seconds of CPU time were spent by each of layers 1 and 2, and 80 seconds were spent by layer 3 (Fig. 10(b)).

Again, as shown in Theorems 2 and 3, unimportant parts of the training data can be safely ignored when layer 2 is learning. In this experiment, at time $t = 60,000$ (which is the final time step of layer 2's learning period), $92.2 \pm 0.6\%$ of the training data were safely ignored. Furthermore, as shown in Theorem 4, more of the training data can be safely ignored when layer 3 is learning. In this experiment, at time $t = 100,000$ (which is the final time step of layer 3's learning period), $99.2 \pm 0.1\%$ of the training data were safely ignored.

Here, we used two "tricks" that made learning more likely to succeed. Both focus on the learning of layer 2. The first trick is that, when we carry out the greedy optimization of U^2 and P^2, we try to predict only observations that occur within the building; when the observation is a building number (which occurs when the agent moves from one building to another), we ignore it by replacing the observation with "no move." Using this trick, layer 2 can focus on learning the map within each building, which indeed resulted in better overall prediction performance after the learning of layer 3.

The second trick is to try to find M^2 such that the number of possible configurations of $(b^1(t), b^2(t), a(t))$ is minimized. If there are a large number of configurations of $(b^1(t), b^2(t), a(t))$, the learning of layer 3 is difficult. Thus, keeping the number of configurations small is important. To find such M^2, we compared results of the greedy optimization of U^2 and P^2 (recall that we obtained 10 optimization results using different initial values), and if there were multiple optimization results with the same prediction performance, we picked the one that caused the minimum number of configurations of $(b^1(t), b^2(t), a(t))$.

With these two tricks, perfect prediction was achieved in 20 out of 20 runs; hence, the average log-likelihood becomes 0 in Fig. 10(a). Without the first trick, perfect prediction was achieved in 10 out of 20 runs. Without the second trick, it was achieved in 16 out of 20 runs. With neither of the tricks, it was achieved in 10 out of 20 runs.

6. Conclusion

In this paper, we proposed a bottom-up learning algorithm for deterministic POMDPs and proved that the algorithm learns a perfect model in a class of POMDP environments, i.e., deterministic POMDPs without ao-loops. We also empirically demonstrated how a hierarchical model is learned via the algorithm.

One key improvement for future study is to extend our learning algorithm to be applicable to more general POMDPs. To make it useful for environments that have ao-loops, one possible approach is to use a loop-detection method. To make the algorithm applicable to probabilistic POMDPs, one possible approach is to consider learning approximate models; notions such as *approximately OP transitions* may be useful.

Another important focus for future study is to consider how to efficiently optimize the policy for the hierarchical model learned by our algorithm. Although general POMDP solvers could be used for optimization, there may be a better solver that exploits the fact that full observability is partly recovered in each layer of the model.

Acknowledgment

We would like to thank the reviewers for their valuable comments. Hideaki Itoh wishes to thank Prof. Kiyohiko Nakamura, Prof. Deb Roy, and Minseok Kim for their helpful suggestions. This study was partially supported by the Ministry of Education, Culture, Sports, Science and Technology in Japan, Grant-in-Aid for Scientific Research (C) 24500277.

References

Åström, K.J. (1965). Optimal control of Markov decision processes with incomplete state estimation, *Journal of Mathematical Analysis and Applications* **10**(1): 174–205.

Barto, A.G. and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning, *Discrete Event Dynamic Systems* **13**(4): 341–379.

Bonet, B. (2009). Deterministic POMDPs revisited, *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI), Montreal, Canada*, pp. 59–66.

Bui, H.H., Phung, D.Q. and Venkatesh, S. (2004). Hierarchical hidden Markov models with general state hierarchy, *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI), San Jose, CA, USA*, pp. 324–329.

Chang, H.S., Fard, P.J., Marcus, S.I. and Shayman, M. (2003). Multi-time scale Markov decision processes, *IEEE Transactions on Automatic Control* **48**(6): 976–987.

Charlin, L., Poupart, P. and Shioda, R. (2007). Automated hierarchy discovery for planning in partially observable environments, *in* B. Schölkopf, J.C. Platt and T. Hofmann (Eds.), *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, The MIT Press, Cambridge, MA, pp. 225–232.

Chatzis, S.P. and Kosmopoulos, D. (2014). A partially-observable Markov decision process for dealing with dynamically changing environments, *in* L. Iliadis, I. Maglogiannis and H. Papadopoulos (Eds.), *Artificial Intelligence Applications and Innovations*, Springer-Verlag, Berlin, pp. 111–120.

Dean, T., Angluin, D., Basye, K., Engelson, S., Kaelbling, L., Kokkevis, E. and Maron, O. (1995). Inferring finite automata with stochastic output functions and an application to map learning, *Machine Learning* **18**(1): 81–108.

Dietterich, T.G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition, *Journal of Artificial Intelligence Research* **13**: 227–303.

Doshi-Velez, F. (2009). The infinite partially observable Markov decision process, *in* Y. Bengio *et al.* (Eds.), *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, Curran Associates Inc., Red Hook, NY, pp. 477–485.

Doshi-Velez, F., Pfau, D., Wood, F. and Roy, N. (2015). Bayesian nonparametric methods for partially-observable reinforcement learning, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **37**(2): 394–407.

Drake, A. (1962). *Observation of a Markov Process Through a Noisy Channel*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Fine, S., Singer, Y. and Tishby, N. (1998). The hierarchical hidden Markov model: Analysis and applications, *Machine Learning* **32**(1): 41–62.

Foka, A. and Trahanias, P. (2007). Real-time hierarchical POMDPs for autonomous robot navigation, *Robotics and Autonomous Systems* **55**(7): 561–571.

Gavaldà, R., Keller, P.W., Pineau, J. and Precup, D. (2006). PAC-learning of Markov models with hidden state, *Proceedings of the 17th European Conference on Machine Learning (ECML), Berlin, Germany*, pp. 150–161.

Heller, K.A., Teh, Y.W. and Görür, D. (2009). Infinite hierarchical hidden Markov models, *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS), Clearwater Beach, FL, USA*, pp. 224–231.

Hengst, B. (2011). Hierarchical approaches, *in* M. Wiering and M. van Otterlo (Eds.), *Reinforcement Learning: State of the Art*, Springer-Verlag, Berlin, pp. 293–323.

Hinton, G.E., Osindero, S. and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets, *Neural Computation* **18**(7): 1527–1554.

Hoey, J., Poupart, P., Bertoldi, A., Craig, T., Boutilier, C. and Mihailidis, A. (2010). Automated handwashing assistance for persons with dementia using video and a partially observable Markov decision process, *Computer Vision and Image Understanding* **114**(5): 503–519.

Holmes, M.P. and Isbell Jr., C.L. (2006). Looping suffix tree-based inference of partially observable hidden state, *Proceedings of the 23rd International Conference on Machine Learning (ICML), Pittsburgh, PA, USA*, pp. 409–416.

Kaelbling, L.P., Littman, M.L. and Cassandra, A.R. (1999). Planning and acting in partially observable stochastic domains, *Artificial Intelligence* **101**(1–2): 99–134.

Kolobov, A. (2012). Planning with Markov decision processes: An AI perspective, *Synthesis Lectures on Artificial Intelligence and Machine Learning* **6**(1): 1–210.

Kołodziej, J., Khan, S.U., Wang, L., Min-Allah, N., Madani, S.A., Ghani, N. and Li, H. (2011). An application of Markov jump process model for activity-based indoor mobility prediction in wireless networks, *9th IEEE International Conference on Frontiers of Information Technology (FIT), Islamabad, Pakistan*, pp. 51–56.

Li, H., Zhao, Q. and Yang, Z. (2007). Reliability modeling of fault tolerant control systems, *International Journal of Applied Mathematics and Computer Science* **17**(4): 491–504, DOI: 10.2478/v10006-007-0041-0.

Lim, Z., Sun, L. and Hsu, D.J. (2011). Monte Carlo value iteration with macro-actions, *in* J. Shawe-Taylor *et al.* (Eds.), *Advances in Neural Information Processing Systems 24 (NIPS 2011)*, Curran Associates Inc., Red Hook, NY, pp. 1287–1295.

Littman, M.L. (1996). *Algorithms for Sequential Decision Making*, Ph.D. thesis, Brown University, Providence, RI.

Mahadevan, S. (1998). Partially observable semi-Markov decision processes: Theory and applications in engineering and cognitive science, *AAAI Fall Symposium on Planning with Partially Observable Markov Decision Processes, Orlando, FL, USA*, pp. 113–120.

Mihalkova, L. and Mooney, R.J. (2007). Bottom-up learning of Markov logic network structure, *Proceedings of the 24th International Conference on Machine Learning (ICML), Corvallis, OR, USA*, pp. 625–632.

Murphy, K.P. (2002). Representing hierarchical POMDPs as DBNs, with applications to mobile robot navigation, `www.cs.ubc.ca/~murphyk/mypapers.html`.

Oliver, N., Garg, A. and Horvitz, E. (2004). Layered representations for learning and inferring office activity from multiple sensory channels, *Computer Vision and Image Understanding* **96**(2): 163–180.

Oniszczuk, W. (2009). Semi-Markov-based approach for the analysis of open tandem networks with blocking and truncation, *International Journal of Applied Mathematics and Computer Science* **19**(1): 151–163, DOI: 10.2478/v10006-009-0014-6.

Pineau, J., Montemerlo, M., Pollack, M., Roy, N. and Thrun, S. (2003). Towards robotic assistants in nursing homes: Challenges and results, *Robotics and Autonomous Systems* **42**(3): 271–281.

Poupart, P. and Vlassis, N. (2008). Model-based Bayesian reinforcement learning in partially observable domains, *Proceedings of the 10th International Symposium on Artificial Intelligence and Mathematics (ISAIM), Fort Lauderdale, FL, USA,* p. 8.

Rao, V. and Teh, Y. W. (2013). Fast MCMC sampling for Markov jump processes and extensions, *The Journal of Machine Learning Research* **14**(1): 3295–3320.

Ross, S., Pineau, J., Chaib-draa, B. and Kreitmann, P. (2011). A Bayesian approach for learning and planning in partially observable Markov decision processes, *Journal of Machine Learning Research* **12**: 1729–1770.

Roy, N., Pineau, J. and Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning, *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL), Hong Kong, China*, pp. 93–100.

Rusek, K., Janowski, L. and Papir, Z. (2014). Transient and stationary characteristics of a packet buffer modelled as an $MAP/SM/1/b$ system, *International Journal of Applied Mathematics and Computer Science* **24**(2): 429–442, DOI: 10.2478/amcs-2014-0033.

Sallans, B. (2000). Learning factored representations for partially observable Markov decision processes, *in* S.A. Solla, T.K. Leen and K. Müller (Eds.), *Advances in Neural Information Processing Systems 12 (NIPS 1999)*, The MIT Press, Cambridge, MA, pp. 1050–1056.

Shani, G., Brafman, R.I. and Shimony, S.E. (2005). Model-based online learning of POMDPs, *Proceedings of the 16th European Conference on Machine Learning (ECML), Porto, Portugal*, pp. 353–364.

Spaan, M.T.J. and Vlassis, N. (2005). Perseus: Randomized point-based value iteration for POMDPs, *Journal of Artificial Intelligence Research* **24**: 195–220.

Theocharous, G. (2002). *Hierarchical Learning and Planning in Partially Observable Markov Decision Processes*, Ph.D. thesis, Michigan State University, East Lansing, MI.

Theocharous, G. and Mahadevan, S. (2002). Approximate planning with hierarchical partially observable Markov decision process models for robot navigation, *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA), Washington, DC, USA,* pp. 1347–1352.

Theocharous, G., Murphy, K. and Kaelbling, L.P. (2004). Representing hierarchical POMDPs as DBNs for multi-scale robot localization, *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA), New Orleans, LA, USA*, Vol. 1, pp. 1045–1051.

Toussaint, M., Charlin, L. and Poupart, P. (2008). Hierarchical POMDP controller optimization by likelihood maximization, *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI), Helsinki, Finland*, pp. 562–570.

Wakabayashi, K. and Miura, T. (2012). Forward-backward activation algorithm for hierarchical hidden Markov models, *in* F. Pereira *et al.* (Eds.), *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, Curran Associates Inc., Red Hook, NY, pp. 1502–1510.

White, C.C. (1976). Procedures for the solution of a finite-horizon, partially observed, semi-Markov optimization problem, *Operations Research* **24**(2): 348–358.

Young, S., Gašić, M., Thomson, B. and Williams, J.D. (2013). POMDP-based statistical spoken dialog systems: A review, *Proceedings of the IEEE* **101**(5): 1160–1179.

Youngblood, G.M. and Cook, D.J. (2007). Data mining for hierarchical model creation, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **37**(4): 561–572.

Youngblood, G.M., Heierman, E.O., Cook, D.J. and Holder, L.B. (2005). Automated HPOMDP construction through data-mining techniques in the intelligent environment domain, *Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference (FLAIRS), Clearwater Beach, FL, USA*, pp. 194–199.

Zamani, Z., Sanner, S., Poupart, P. and Kersting, K. (2012). Symbolic dynamic programming for continuous state and observation POMDPs, *in* F. Pereira *et al.* (Eds.), *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, Curran Associates Inc., Red Hook, NY, pp. 1403–1411.

Hideaki Itoh received the B.E., M.E., and D.E. degrees in 1997, 1999, and 2002, respectively, from the University of Tokyo, Japan. After working as a research associate and as an assistant professor at Tokyo Institute of Technology, he transferred as a lecturer to the Department of Electrical and Electronic Engineering at Saga University in 2009. His research interests include artificial intelligence and soft computing.

Hisao Fukumoto received the B.E., M.E., and D.E. degrees in 1997, 1999, and 2002, respectively, from Saga University, Japan. He was a research associate of the Faculty of Science and Engineering at Saga University in 2002–2007. He has been an assistant professor of the Graduate School of Science and Engineering at Saga University since 2007. His research fields include image processing, signal processing, and higher education on engineering based on advanced computational engineering.

Hiroshi Wakuya received the B.E. degree in 1989 from Kyushu Institute of Technology, Kitakyushu, Japan. He obtained the M.E. and D.E. degrees in 1991 and 1994, respectively, from Tohoku University, Sendai, Japan. He joined Saga University, Japan, as a research associate in 1994. He has been an associate professor of the Department of Electrical and Electronic Engineering at Saga University since 2002. His research fields cover biomedical engineering and soft computing.

Tatsuya Furukawa received the B.E., M.E., and D.E. degrees in 1979, 1981, and 1987, respectively, from Kyushu University, Fukuoka, Japan. He joined Nagasaki University, Japan, as a research associate in 1984. He transferred to Saga University, Japan, as a lecturer in 1986. He has been a professor at the Department of Electrical and Electronic Engineering at Saga University since 2001. His research fields are power engineering, electric machinery, computer science, simulation technology, instrumentation technology, and higher education on engineering based on advanced computational engineering.

Appendix

In this section, we provide proofs of Theorems 2–4. For brevity, we write M^1's belief and belief space as b and B, instead of b^1 and B^1, respectively.

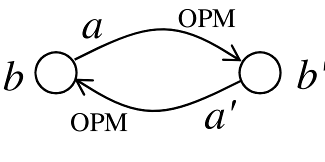
A1. Proof of Theorem 2

Before proving Theorem 2, we begin with a lemma regarding bidirectionality of the OPM transitions, as illustrated in Fig. A1.

Lemma A1. *If (b, a) causes an OPM transition to b', then there exists an action a' such that (b', a') causes an OPM transition returning to b.*

Proof. Suppose that (b, a) causes an OPM transition to b'. Then, by the definition of OPM transitions (Definition 9), (b, a) causes an OP transition to b', and b and b' are mutually observation-predictably transitionable. Therefore, by the definition of being mutually observation-predictably transitionable (Definition 8), there exists at least one action a' such that (b', a') causes an OP transition returning to b. This OP transition is an OPM transition because b and b' are mutually observation-predictably transitionable. Therefore, there exists an action a' such that (b', a') causes an OPM transition to b. ■

Next, we need a definition and a lemma regarding the extended states. We use the term *OPM transition* in the following broader sense: For any action a and two extended states $x = (s, b)$ and $x' = (s', b')$ such that x transitions to x' after action a, we say that (x, a) causes an OPM transition to x' when (b, a) causes an OPM transition to b'.



$$\begin{tikzcd} b \arrow[r, bend left, "a\ \text{OPM}"] & b' \arrow[l, bend left, "\text{OPM}\ a'"] \end{tikzcd}$$

Fig. A1. If we have $(b, a) \rightarrow_{\text{OPM}} b'$, then there exists a' such that $(b', a') \rightarrow_{\text{OPM}} b$.

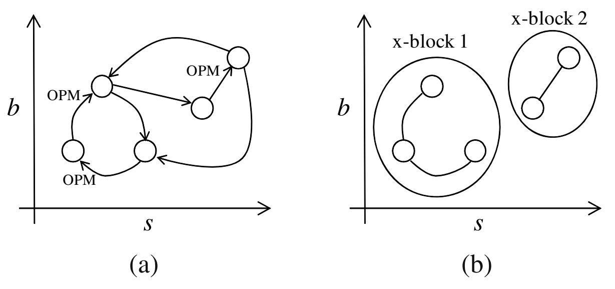


Fig. A2. Partitioning the space of extended states (s, b) into x-blocks. Extended states (◯) are connected by transitions ($\rightarrow$); some transitions are OPM ($\rightarrow_{\text{OPM}}$), while the others are not (a). Partitioned extended states (b).

Definition A1. (*x-block*) Let us consider any strongly connected region of the combined POMDP consisting of any environment E and any FSM M^1. By drawing all extended states in the strongly connected region and all transitions between these extended states caused by every action $a \in A$, we obtain a transition diagram such as the one shown in Fig. A2(a). From these transitions, let us keep the OPM transitions and delete all the other transitions. Let us ignore the directions of the OPM transitions. Then, we obtain a connection diagram such as the one shown in Fig. A2(b). In this connection diagram, the extended states are partitioned into blocks; within each block, all of the extended states are connected by the undirected OPM transitions. We call each block an *extended-block*, or *x-block* for short.

Lemma A2. *Consider any environment E without ao-loops and any FSM M^1. For any two extended states (s, b) and (s', b') within any x-block of the combined POMDP consisting of E and M^1, we have $s \neq s' \Rightarrow b \neq b'$.*

Proof. We prove this by contradiction. Assume that, in an x-block of the combined POMDP, there are two extended states,

$$x_1 = (s_1, b) \text{ and } x_2 = (s_2, b), \text{ where } s_1 \neq s_2. \quad \text{(A1)}$$

Note that x_1 and x_2 have different states s_1 and $s_2 \in S$ and common belief $b \in B$.

First, let us consider the case in which there is a path or an action-observation sequence q that we can follow from x_1 to x_2 via OPM transitions (Fig. A3).

Let the sequence of actions along path q be denoted by q_a, and the resultant observation sequence by q_o. That is, if $q = (a_1, o_1, a_2, o_2, \ldots, a_n, o_n)$, then $q_a = (a_1, a_2, \ldots, a_n)$ and $q_o = (o_1, o_2, \ldots, o_n)$. Suppose that the agent in x_2 takes the same action sequence q_a. Then, by the definition of OPM transitions, the same observation sequence q_o is obtained; hence, the agent goes to some

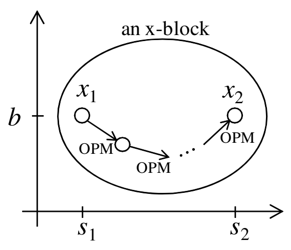


Fig. A3. Case in which there is a path that we can follow from x_1 to x_2 by OPM transitions, where x_1 and x_2 are defined in Eqn. (A1).

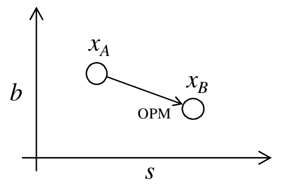


Fig. A4. OPM transition from x_A to x_B.

extended state whose belief is b. We denote this extended state as $x_3 = (s_3, b)$.

Here, x_3 cannot be identical to x_2, because if $s_3 = s_2$, then it is implied that a 1st-order ao-loop $s_1 \rightarrow_q s_2 \rightarrow_q s_2$ exists, and this contradicts the supposition that environment E has no ao-loops. Similarly, x_3 cannot be identical to x_1, because if $s_3 = s_1$, it is implied that a 2nd-order ao-loop $s_1 \rightarrow_q s_2 \rightarrow_q s_1$ exists, which again is a contradiction. Thus, x_3 is not identical to either x_1 or x_2.

Therefore, x_1, x_2, and x_3 are all different from one another. Suppose that the agent in x_3 takes the same action sequence q_a. Then, again by the definition of OPM transitions, the same observation sequence q_o is obtained, and the agent goes to some extended state $x_4 = (s_4, b)$. Here, x_4 cannot be identical to x_1, because if $s_4 = s_1$, it is implied that a 3rd-order ao-loop exists, which is a contradiction. Similarly, x_4 cannot be identical to x_2 or x_3. Thus, x_4 is not identical to any of x_1, x_2, or x_3.

Therefore, x_1, x_2, x_3, and x_4 are all different from one another. After repeating this, we find that $x_1, x_2, x_3, \ldots,$ and $x_{|S|+1}$ (all of them have common belief b) are all different from one another, where $|S|$ is the number of states of environment E. However, since there are only $|S|$ states in environment E, it is impossible that $x_1, x_2, x_3, \ldots,$ and $x_{|S|+1}$ are all different.

Therefore, we conclude that there is no such path that we can follow from x_1 to x_2 by OPM transitions (*).

We can also conclude, however, that there *is* a path that we can follow from x_1 to x_2 by OPM transitions. To prove this, we begin by proving that, for any two different extended states x_A and x_B, if there is an OPM transition from x_A to x_B (as illustrated in Fig. A4), then there is an

OPM transition from x_B back to x_A.

The proof is as follows. Suppose that there is an OPM transition from $x_A = (s_A, b_A)$ to $x_B = (s_B, b_B)$. It follows that there is an action a such that (b_A, a) causes an OPM transition to b_B. Thus, from Lemma A1, there exists an action a' such that (b_B, a') causes an OPM transition to b_A. Therefore, there is at least one OPM transition from x_B to an extended state whose belief is b_A. We denote this extended state as $x_C = (s_C, b_A)$. Here, if $x_C \neq x_A$, then we can go from x_A to $x_C (\neq x_A)$ by OPM transitions through x_B; however, going from x_A to $x_C (\neq x_A)$ by OPM transitions is impossible, which can be proven using the same technique as for (*). Thus, we have $x_C = x_A$, which means that there is an OPM transition from x_B to x_A. Therefore, we have proven that if there is an OPM transition from x_A to x_B, then there is an OPM transition from x_B to x_A.

Thus, OPM transitions between extended states are bidirectional. From the definition of x-blocks, we know that any two extended states within an x-block are connected by OPM transitions whose directions are ignored. Because of this and the bidirectionality of OPM transitions, we conclude that there is a path that we can follow from x_1 to x_2 by OPM transitions (**).

Since (*) and (**) contradict each other, our initial assumption is false. ■

Given this lemma, it is easy to prove Theorem 2, which we restate here, omitting superscripts of b^1 and B^1.

Theorem A1. (Theorem 2 restated) *Consider a combined POMDP consisting of any environment E without ao-loops and any FSM M^1. After the agent falls into a strongly connected region of $S \times B$ of the combined POMDP, if there is any time period $t_1 \leq t \leq t_2$ such that every $bao(t)$, $t \in [t_1, t_2]$, is associated with an OPM transition, then for any t_i and $t_j \in [t_1, t_2 + 1]$ we have $s(t_i) \neq s(t_j) \Rightarrow b(t_i) \neq b(t_j)$.*

Proof. Since every $bao(t)$, $t \in [t_1, t_2]$, is associated with an OPM transition, the extended state remains located within a single x-block from $t = t_1$ to $t_2 + 1$. Therefore, from Lemma A2, we know that for any t_i and $t_j \in [t_1, t_2 + 1]$, if $s(t_i) \neq s(t_j)$, then $b(t_i) \neq b(t_j)$. ■

A2. Proof of Theorem 3

Here we restate Theorem 3, omitting superscripts of b^1 and B^1.

Theorem A2. (Theorem 3 restated) *Consider any environment E without ao-loops and any FSM M^1 that correctly predicts $o(t + 1)$ whenever $(b(t), a(t))$ causes an OPM transition. For any time t after the agent falls into a strongly connected region of $S \times B$ of the combined POMDP consisting of E and M^1, there exists an $M^2 = \langle B^2, A^2, O, U^2, P^2, b^2_0 \rangle$ that performs perfect prediction at time t, collaborating with M^1 in the manner described in Algorithm 2.*

Proof. Consider any strongly connected region of $S \times B$ of the combined POMDP consisting of any environment E without ao-loops and any FSM M^1. Since S and B are both finite sets, there are a finite number of x-blocks in this region. Let the number of x-blocks be N. Next, let us arbitrarily assign a number from 1 to N to each of the x-blocks, avoiding repetition. Then, we can enumerate the x-blocks as the 1st x-block, 2nd x-block, . . ., i-th x-block, . . ., and N-th x-block. In this proof, we call the assigned number the *index* of the x-block, and we use i as the variable that represents the index.

Consider a function f that maps extended state (s, b) to (i, b), where i is the index of the x-block in which (s, b) is located. This function f is invertible; i.e., there exists f^{-1} such that $f^{-1}(i, b) = (s, b)$. Invertibility holds because for any x-block i and any belief b, we know from Lemma A2 that inside the i-th x-block there is only one state s that can co-occur with b.

Using f, we can construct an M^2 that performs perfect prediction at time t in collaboration with M^1, where t is any time after the agent falls into the strongly connected region of $S \times B$. In what follows, we describe how to construct such M^2. Note that the M^2 which we construct below is such that its belief b^2 is always identical to the index of the x-block in which the agent is located.

First, let B^2 be the set of indices $\{1, 2, \ldots, N\}$. As stated in Section 3.3.3, let A^2 be the set of possible values of (b, a), and O be the set of possible observations. Let t^2_0 (i.e., the initial time of M^2) be the time to perform perfect prediction, and let b^2_0 be the index of the x-block in which the agent is located at time t^2_0.

Next, let us construct U^2 and P^2 as follows. Suppose that, at time τ $(\geq t^2_0)$, the agent is in the i-th x-block with its layer 1's belief being b. The state s at time τ is uniquely determined by $f^{-1}(i, b)$.

The first case we consider is when the agent takes action a such that (b, a) does not cause an OPM transition. After action a is taken, the state s is changed to $s' = \delta(s, a)$, and the agent observes $o' = \gamma(s, a)$, by which layer 1's belief b is changed to $b' = U^1(b, a, o')$. Thus, both s' and b' are uniquely determined from i, b, and a. Therefore, the new x-block i' in which the agent is located after taking action a is also uniquely determined from i, b, and a. This implies that there exists a function U^2 that maps $(i, (b, a), o')$ to i'. With this U^2, layer 2's belief b^2 is always identical to the index of the x-block in which the agent is located. In addition, if we set P^2 to be the function that maps i and (b, a) to o', then observation o' is correctly predicted by P^2.

The other case is when the agent takes action a such that (b, a) causes an OPM transition. In this case, by definition, the agent remains in the same i-th x-block.

Since M^2's belief b^2 is not changed (i.e., Step (2a) of Algorithm 2), if b^2 is identical to i before taking the action, then b^2 remains identical to i after the action. The fact that layer 2 does not make a prediction causes no problem because observation o' is correctly predicted by layer 1.

Thus, we can construct at least one M^2 by which perfect prediction is achieved. ■

A3. Proof of Theorem 4

Proof. Consider the case when $l = 2$. In this proof, we denote each value of $b^{1:2}ao(t)$ as b^1, b^2, a, and o; i.e., we let $(b^1, b^2) := b^{1:2}(t), a := a(t)$, and $o := o(t)$. Suppose that $b^1ao(t)$, or equivalently (b^1, a, o), is associated with an OPM transition. Let M^1's belief after the transition be $b^{1\prime} = U^1(b^1, a, o)$.

First, since M^2's belief b^2 is not changed when (b^1, a) causes an OPM transition (i.e., Step (2a) of Algorithm 2), and since every OPM transition is an OP transition, we know that $((b^1, b^2), a)$ causes an OP transition to $(b^{1\prime}, b^2)$ (*).

Next, we know from Lemma A1 that, since (b^1, a) causes an OPM transition to $b^{1\prime}$, there exists at least one action a' such that $(b^{1\prime}, a')$ causes an OPM transition to b^1. Since M^2's belief b^2 is not changed when $(b^{1\prime}, a')$ causes an OPM transition, and since every OPM transition is an OP transition, we know that there exists at least one action a' such that $((b^{1\prime}, b^2), a')$ causes an OP transition to (b^1, b^2). Given this and (*), we know that (b^1, b^2) and $(b^{1\prime}, b^2)$ are mutually observation-predictably transitionable (**).

Because of (*) and (**), we know that $b^{1:2}ao(t)$ is associated with an OPM transition. Thus, the theorem is proven for the case of $l = 2$.

For $l > 2$, we can prove the theorem in the same way as for $l = 2$, except that we need to use $M^{1:l-1}$ instead of M^1, where $M^{1:l-1}$ is a single FSM that works equivalently to layers $M^1, M^2, \ldots$, and M^{l-1} (see Section 4.3 for details regarding this technique). ■

Received: 24 April 2014
Revised: 29 November 2014
Re-revised: 30 January 2015