Chapter 12
Partially Observable Markov Decision Processes

Matthijs T.J. Spaan

Abstract. For reinforcement learning in environments in which an agent has access to a reliable state signal, methods based on the Markov decision process (MDP) have had many successes. In many problem domains, however, an agent suffers from limited sensing capabilities that preclude it from recovering a Markovian state signal from its perceptions. Extending the MDP framework, partially observable Markov decision processes (POMDPs) allow for principled decision making under conditions of uncertain sensing. In this chapter we present the POMDP model by focusing on the differences with fully observable MDPs, and we show how optimal policies for POMDPs can be represented. Next, we give a review of model-based techniques for policy computation, followed by an overview of the available model-free methods for POMDPs. We conclude by highlighting recent trends in POMDP reinforcement learning.

12.1 Introduction

The Markov decision process model has proven very successful for learning how to act in stochastic environments. In this chapter, we explore methods for reinforcement learning by relaxing one of the limiting factors of the MDP model, namely the assumption that the agent knows with full certainty the state of the environment. Put otherwise, the agent's sensors allow it to perfectly monitor the state at all times, where the state captures all aspects of the environment relevant for optimal decision making. Clearly, this is a strong assumption that can restrict the applicability of the MDP framework. For instance, when certain state features are hidden from

Matthijs T.J. Spaan
Institute for Systems and Robotics, Instituto Superior Técnico,
Av. Rovisco Pais 1, 1049-001 Lisbon, Portugal,
(Currently at Delft University of Technology, Delft, The Netherlands)
e-mail: mtjspaan@isr.ist.utl.pt

M. Wiering and M. van Otterlo (Eds.): Reinforcement Learning, ALO 12, pp. 387–414.
springerlink.com

the agent the state signal will no longer be Markovian, violating a key assumption of most reinforcement-learning techniques (Sutton and Barto, 1998).[1]

One example of particular interest arises when applying reinforcement learning to embodied agents. In many robotic applications the robot's on-board sensors do not allow it to unambiguously identify its own location or pose (Thrun et al, 2005). Furthermore, a robot's sensors are often limited to observing its direct surroundings, and might not be adequate to monitor those features of the environment's state beyond its vicinity, so-called hidden state. Another source of uncertainty regarding the true state of the system are imperfections in the robot's sensors. For instance, let us suppose a robot uses a camera to identify the person it is interacting with. The face-recognition algorithm processing the camera images is likely to make mistakes sometimes, and report the wrong identity. Such an imperfect sensor also prevents the robot from knowing the true state of the system: even if the vision algorithm reports person A, it is still possible that person B is interacting with the robot. Although in some domains the issues resulting from imperfect sensing might be ignored, in general they can lead to severe performance deterioration (Singh et al, 1994).

Instead, in this chapter we consider an extension of the (fully observable) MDP setting that also deals with uncertainty resulting from the agent's imperfect sensors. A partially observable Markov decision process (POMDP) allows for optimal decision making in environments which are only partially observable to the agent (Kaelbling et al, 1998), in contrast with the full observability mandated by the MDP model. In general the partial observability stems from two sources: (i) multiple states give the same sensor reading, in case the agent can only sense a limited part of the environment, and (ii) its sensor readings are noisy: observing the same state can result in different sensor readings. The partial observability can lead to "perceptual aliasing": different parts of the environment appear similar to the agent's sensor system, but require different actions. The POMDP captures the partial observability in a probabilistic observation model, which relates possible observations to states.

Classic POMDP examples are the machine maintenance (Smallwood and Sondik, 1973) or structural inspection (Ellis et al, 1995) problems. In these types of problems, the agent has to choose when to inspect a certain machine part or bridge section, to decide whether maintenance is necessary. However, to allow for inspection the machine has to be stopped, or the bridge to be closed, which has a clear economic cost. A POMDP model can properly balance the trade-off between expected deterioration over time and scheduling inspection or maintenance activities. Furthermore, a POMDP can model the scenario that only choosing to inspect provides information regarding the state of the machine or bridge, and that some flaws are not always revealed reliably. More recently, the POMDP model has gained in relevance for robotic applications such as robot navigation (Simmons and Koenig, 1995; Spaan and Vlassis, 2004; Roy et al, 2005; Foka and Trahanias, 2007), active sensing (Hoey and Little, 2007; Spaan et al, 2010), object grasping (Hsiao et al, 2007) or human-robot interaction (Doshi and Roy, 2008). Finally, POMDPs have been

[1] Note to editor: this point is most likely mentioned before in the book, for reasons of coherence this citation can be replaced with a reference to the correct section.

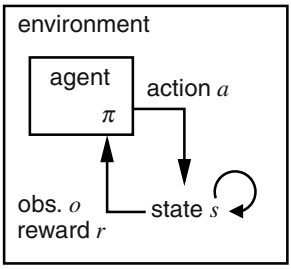


Fig. 12.1 A POMDP agent interacting with its environment

applied in diverse domains such as treatment planning in medicine (Hauskrecht and Fraser, 2000), spoken dialogue systems (Williams and Young, 2007), developing navigation aids (Stankiewicz et al, 2007), or invasive species management (Haight and Polasky, 2010).

The remainder of this chapter is organized as follows. First, in Section 12.2 we formally introduce the POMDP model, and we show that the partial observability leads to a need for memory or internal state on the part of the agent. We discuss how optimal policies and value functions are represented in the POMDP framework. Next, Section 12.3 reviews model-based techniques for POMDPs, considering optimal, approximate and heuristic techniques. Section 12.4 gives an overview of the model-free reinforcement learning techniques that have been developed for or can be applied to POMDPs. Finally, Section 12.5 describes some recent developments in POMDP reinforcement learning.

12.2 Decision Making in Partially Observable Environments

In this section we formally introduce the POMDP model and related decision-making concepts.

12.2.1 POMDP Model

A POMDP shares many elements with the fully observable MDP model as described in Section 1.3, which we will repeat for completeness. Time is discretized in steps, and at the start of each time step the agent has to execute an action. We will consider only discrete, finite, models, which are by far the most commonly used in the POMDP literature given the difficulties involved with solving continuous models. For simplicity, the environment is represented by a finite set of states $S = \{s^1, \ldots, s^N\}$. The set of possible actions $A = \{a^1, \ldots, a^K\}$ represent the possible ways the agent can influence the system state. Each time step the agent takes an action a in state s, the environment transitions to state s' according to the probabilistic transition function $T(s,a,s')$ and the agent receives an immediate reward $R(s,a,s')$.

What distinguishes a POMDP from a fully observable MDP is that the agent now perceives an observation $o \in \Omega$, instead of observing s' directly. The discrete set of observations $\Omega = \{o^1, \ldots, o^M\}$ represent all possible sensor readings the agent can receive. Which observation the agent receives depends on the next state s' and may also be conditional on its action a, and is drawn according to the observation function $O : S \times A \times \Omega \to [0,1]$. The probability of observing o in state s' after executing a is $O(s',a,o)$. In order for O to be a valid probability distribution over possible observations it is required that $\forall s' \in S, a \in A, o \in \Omega \; O(s',a,o) \geq 0$ and that $\sum_{o \in \Omega} O(s',a,o) = 1$. Alternatively, the observation function can also be defined as $O : S \times \Omega \to [0,1]$ reflecting domains in which the observation is independent of the last action.[2]

As in an MDP, the goal of the agent is to act in such a way as to maximize some form of expected long-term reward, for instance

$$E\Big[\sum_{t=0}^{h} \gamma^t R_t\Big], \tag{12.1}$$

where $E[\cdot]$ denotes the expectation operator, h is the planning horizon, and γ is a discount rate, $0 \leq \gamma < 1$.

Analogous to Definition 1.3.1, we can define a POMDP as follows.

Definition 12.2.1. *A **partially observable Markov decision process** is a tuple $\langle S, A, \Omega, T, O, R \rangle$ in which S is a finite set of states, A is a finite set of actions, Ω is a finite set of observations, T is a transition function defined as $T : S \times A \times S \to [0,1]$, O is an observation function defined as $O : S \times A \times \Omega \to [0,1]$ and R is a reward function defined as $R : S \times A \times S \to \mathbb{R}$.*

Fig. 12.1 illustrates these concepts by depicting a schematic representation of a POMDP agent interacting with the environment.

To illustrate how the observation function models different types of partial observability, consider the following examples, which assume a POMDP with 2 states, 2 observations, and 1 action (omitted for simplicity). The case that sensors make mistakes or are noisy can be modeled as follows. For instance,

$$O(s^1,o^1) = 0.8,\ O(s^1,o^2) = 0.2,\ O(s^2,o^1) = 0.2,\ O(s^2,o^2) = 0.8,$$

models an agent equipped with a sensor that is correct in 80% of the cases. When the agent observes o^1 or o^2, it does not know for sure that the environment is in state s^1 resp. s^2. The possibility that the state is completely hidden to the agent can be modeled by assigning the same observation to both states (and observation o^2 is effectively redundant):

$$O(s^1,o^1) = 1.0,\ O(s^1,o^2) = 0.0,\ O(s^2,o^1) = 1.0,\ O(s^2,o^2) = 0.0.$$

[2] Technically speaking, by including the last action taken as a state feature, observation functions of the form $O(s',o)$ can express the same models compared to $O(s',a,o)$ functions.

When the agent receives observation o^1 it is not able to tell whether the environment is in state s^1 or s^2, which models the hidden state adequately.

12.2.2 Continuous and Structured Representations

As mentioned before, the algorithms presented in this chapter operate on discrete POMDPs, in which state, action, and observation spaces can be represented by finite sets. Here we briefly discuss work on continuous as well as structured POMDP representations, which can be relevant for several applications.

Many real-world POMDPs are more naturally modeled using continuous models (Porta et al, 2006; Brunskill et al, 2008), for instance a robot's pose is often described by continuous (x,y,θ) coordinates. Standard solution methods such as value iteration can also be defined for continuous state spaces (Porta et al, 2005), and continuous observation spaces (Hoey and Poupart, 2005) as well as continuous actions (Spaan and Vlassis, 2005b) have been studied. However, beliefs, observation, action and reward models defined over continuous spaces can have arbitrary forms that may not be parameterizable. In order to design feasible algorithms it is crucial to work with models that have simple parameterizations and result in closed belief updates and Bellman backups. For instance, Gaussian mixtures or particle-based representations can be used for representing beliefs and linear combinations of Gaussians for the models (Porta et al, 2006). As an alternative, simulation-based methods are often capable of dealing with continuous state and action spaces (Thrun, 2000; Ng and Jordan, 2000; Baxter and Bartlett, 2001).

Returning to finite models, in many domains a more structured POMDP representation is beneficial compared to a flat representation (in which all sets are enumerated). Dynamic Bayesian networks are commonly used as a factored POMDP representation (Boutilier and Poole, 1996; Hansen and Feng, 2000), in addition to which algebraic decision diagrams can provide compact model and policy representation (Poupart, 2005; Shani et al, 2008). Relational representations, as described in Chapter 8, have also been proposed for the POMDP model (Sanner and Kersting, 2010; Wang and Khardon, 2010). Furthermore, in certain problems structuring the decision making in several hierarchical levels (see Chapter 9) can allow for improved scalability (Pineau and Thrun, 2002; Theocharous and Mahadevan, 2002; Foka and Trahanias, 2007; Sridharan et al, 2010). Finally, in the case when multiple agents are executing a joint task in a partially observable and stochastic environment, the Decentralized POMDP model can be applied (Bernstein et al, 2002; Seuken and Zilberstein, 2008; Oliehoek et al, 2008), see Chapter 15.

12.2.3 Memory for Optimal Decision Making

As the example in Section 12.2.1 illustrated, in a POMDP the agent's observations do not uniquely identify the state of the environment. However, as the rewards

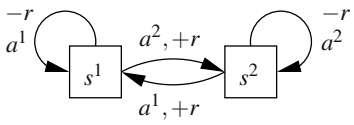


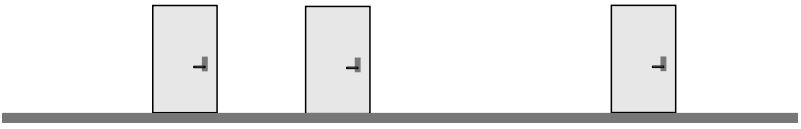
Fig. 12.2 A two-state POMDP from (Singh et al, 1994), in which the agent receives the same observation in both states.

are still associated with the environment state, as well as the state transitions, a single observation is not a Markovian state signal. In particular, a direct mapping of observations to actions is not sufficient for optimal behavior. In order for an agent to choose its actions successfully in partially observable environments memory is needed.

To illustrate this point, consider the two-state infinite-horizon POMDP depicted in Fig. 12.2 (Singh et al, 1994). The agent has two actions, one of which will deterministically transport it to the other state, while executing the other action has no effect on the state. If the agent jumps to the other state it receives a reward of $r > 0$, and $-r$ otherwise. The optimal policy in the underlying MDP has a value of $\frac{r}{1-\gamma}$, as the agent can gather a reward of r at each time step. In the POMDP however, the agent receives the same observation in both states. As a result, there are only two memoryless deterministic stationary policies possible: always execute a^1 or always execute a^2. The maximum expected reward of these policies is $r - \frac{\gamma r}{1-\gamma}$, when the agent successfully jumps to the other state at the first time step. If we allow stochastic policies, the best stationary policy would yield an expected discounted reward of 0, when it chooses either action 50% of the time. However, if the agent could remember what actions it had executed, it could execute a policy that alternates between executing a^1 and a^2. Such a memory-based policy would gather $\frac{\gamma r}{1-\gamma} - r$ in the worst case, which is close to the optimal value in the MDP (Singh et al, 1994).

This example illustrates the need for memory when considering optimal decision making in a POMDP. A straightforward implementation of memory would be to simply store the sequence of actions executed and observations received. However, such a form of memory can grow indefinitely over time, turning it impractical for long planning horizons. Fortunately, a better option exists, as we can transform the POMDP to a belief-state MDP in which the agent summarizes all information about its past using a belief vector $b(s)$ (Stratonovich, 1960; Dynkin, 1965; Åström, 1965). This transformation requires that the transition and observation functions are known to the agent, and hence can be applied only in model-based RL methods.

The belief b is a probability distribution over S, which forms a Markovian signal for the planning task. Given an appropriate state space, the belief is a sufficient statistic of the history, which means the agent could not do any better even if it had remembered the full history of actions and observations. All beliefs are contained in a $(|S|-1)$-dimensional simplex $\Delta(S)$, hence we can represent a belief using $|S|-1$ numbers. Each POMDP problem assumes an initial belief b^0, which for instance can be set to a uniform distribution over all states (representing complete ignorance regarding the initial state of the environment). Every time the agent takes an action a and observes o, its belief is updated by Bayes' rule:



(a) The environment of the robot.

(b) Initial belief at $t = 0$.

(c) Updated belief at $t = 1$ after executing *forward* and observing *door*.

(d) Updated belief at $t = 2$ after executing *forward* and observing *door*.

(e) Updated belief at $t = 3$ after executing *forward* and observing *corridor*.

Fig. 12.3 Belief-update example (adapted from Fox et al (1999)). (a) A robot moves in a one-dimensional corridor with three identical doors. (b)-(e) The evolution of the belief over time, for details see main text.

$$b^{ao}(s') = \frac{p(o|s',a)}{p(o|b,a)} \sum_{s \in S} p(s'|s,a) b(s), \tag{12.2}$$

where $p(s'|s,a)$ and $p(o|s',a)$ are defined by model parameters T resp. O, and

$$p(o|b,a) = \sum_{s' \in S} p(o|s',a) \sum_{s \in S} p(s'|s,a) b(s) \tag{12.3}$$

is a normalizing constant.

Fig. 12.3 shows an example of a sequence of belief updates for a robot navigating in a corridor with three identical doors. The corridor is discretized in 26 states and is circular, i.e., the right end of the corridor is connected to the left end. The robot can observe either *door* or *corridor*, but its sensors are noisy. When the robot is positioned in front of a door, it observes *door* with probability 0.9 (and *corridor* with probability 0.1). When the robot is not located in front of a door the probability of observing *corridor* is 0.9. The robot has two actions, *forward* and *backward* (right

resp. left in the figure), which transport the robot 3 (20%), 4 (60%), or 5 (20%) states in the corresponding direction. The initial belief b^0 is uniform, as displayed in Fig. 12.3b. Fig. 12.3c through (e) show how the belief of the robot is updated as it executes the *forward* action each time. The true location of the robot is indicated by the dark-gray component of its belief. In Fig. 12.3c we see that the robot is located in front of the first door, and although it is fairly certain it is located in front of a door, it cannot tell which one. However, after taking another move forward it again observes *door*, and now can pinpoint its location more accurately, because of the particular configuration of the three doors (Fig. 12.3d). However, in Fig. 12.3e the belief blurs again, which is due to the noisy transition model and the fact that the *corridor* observation is not very informative in this case.

12.2.4 Policies and Value Functions

As in the fully observable MDP setting, the goal of the agent is to choose actions which fulfill its task as well as possible, i.e., to learn an optimal policy. In POMDPs, an optimal policy $\pi^*(b)$ maps beliefs to actions. Note that, contrary to MDPs, the policy $\pi(b)$ is a function over a continuous set of probability distributions over S. A policy π can be characterized by a value function $V^\pi : \Delta(S) \rightarrow \mathbb{R}$ which is defined as the expected future discounted reward $V^\pi(b)$ the agent can gather by following π starting from belief b:

$$V^\pi(b) = E_\pi\Big[\sum_{t=0}^{h} \gamma^t R(b_t,\pi(b_t))\Big| b_0 = b\Big], \tag{12.4}$$

where $R(b_t,\pi(b_t)) = \sum_{s\in S} R(s,\pi(b_t))b_t(s)$.

A policy π which maximizes V^π is called an optimal policy π^*; it specifies for each b the optimal action to execute at the current step, assuming the agent will also act optimally at future time steps. The value of an optimal policy π^* is defined by the optimal value function V^*. It satisfies the Bellman optimality equation

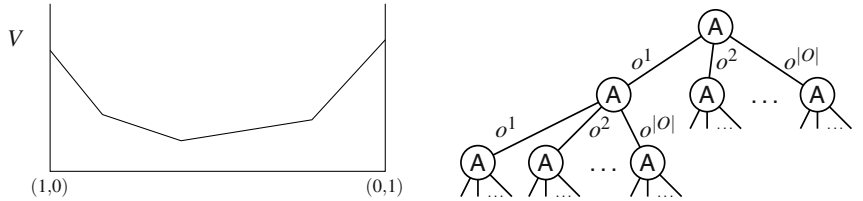
$$V^* = H_{\text{POMDP}} V^*, \tag{12.5}$$

where H_{POMDP} is the Bellman backup operator for POMDPs, defined as:

$$V^*(b) = \max_{a\in A}\Big[\sum_{s\in S} R(s,a)b(s) + \gamma \sum_{o\in O} p(o|b,a)V^*(b^{ao})\Big], \tag{12.6}$$

with b^{ao} given by (12.2), and $p(o|b,a)$ as defined in (12.3). When (12.6) holds for every $b \in \Delta(S)$ we are ensured the solution is optimal.

Computing value functions over a continuous belief space might seem intractable at first, but fortunately the value function has a particular structure that we can exploit (Sondik, 1971). It can be parameterized by a finite number of vectors and has a convex shape. The convexity implies that the value of a belief close to one of the



(a) Example value function. (b) Example policy tree.

Fig. 12.4 (a) An example of a value function in a two-state POMDP. The y-axis shows the value of each belief, and the x-axis depicts the belief space $\Delta(S)$, ranging from $(1,0)$ to $(0,1)$. (b) An example policy tree, where at a node the agent takes an action, and it transitions to a next node based on the received observation $o \in \{o^1, o^2, \ldots, o^{|O|}\}$.

corners of the belief simplex $\Delta(S)$ will be high. In general, the less uncertainty the agent has over its true state, the better it can predict the future, and as such take better decisions. A belief located exactly at a particular corner of $\Delta(S)$, i.e., $b(s) = 1$ for a particular s, defines with full certainty the state of the agent. In this way, the convex shape of V can be intuitively explained. An example of a convex value function for a two-state POMDP is shown in Fig. 12.4a. As the belief space is a simplex, we can represent any belief in a two-state POMDP on a line, as $b(s^2) = 1 - b(s^1)$. The corners of the belief simplex are denoted by $(1,0)$ and $(0,1)$, which have a higher (or equal) value than a belief in the center of the belief space, e.g., $(0.5,0.5)$.

An alternative way to represent policies in POMDPs is by considering *policy trees* (Kaelbling et al, 1998). Fig. 12.4b shows a partial policy tree, in which the agent starts at the root node of tree. Each node specifies an action which the agent executes at the particular node. Next it receives an observation o, which determines to what next node the agent transitions. The depth of the tree depends on the planning horizon h, i.e., if we want the agent to consider taking h steps, the corresponding policy tree has depth h.

12.3 Model-Based Techniques

If a model of the environment is available, it can be used to compute a policy for the agent. In this section we will discuss several ways of computing POMDP policies, ranging from optimal to approximate and heuristic approaches. Even when the full model is known to the agent, solving the POMDP optimally is typically only computationally feasible for small problems, hence the interest in methods that compromise optimality for reasons of efficiency. All the methods presented in this section exploit a belief state representation (Section 12.2.3), as it provides a compact representation of the complete history of the process.

12.3.1 Heuristics Based on MDP Solutions

First, we discuss some heuristic control strategies that have been proposed which rely on a solution $\pi^*_{\text{MDP}}(s)$ or $Q^*_{\text{MDP}}(s,a)$ of the underlying MDP (Cassandra et al, 1996). The idea is that solving the MDP is of much lower complexity than solving the POMDP (P-complete vs. PSPACE-complete) (Papadimitriou and Tsitsiklis, 1987), but by tracking the belief state still some notion of imperfect state perception can be maintained. Cassandra (1998) provides an extensive experimental comparison of MDP-based heuristics.

Perhaps the most straightforward heuristic is to consider for a belief at a given time step its most likely state (MLS), and use the action the MDP policy prescribes for the state

$$\pi_{MLS}(b) = \pi^*_{\text{MDP}}(\arg\max_s b(s)). \tag{12.7}$$

The MLS heuristic completely ignores the uncertainty in the current belief, which clearly can be suboptimal.

A more sophisticated approximation technique is Q_{MDP} (Littman et al, 1995), which also treats the POMDP as if it were fully observable. Q_{MDP} solves the MDP and defines a control policy

$$\pi_{Q_{\text{MDP}}}(b) = \arg\max_a \sum_s b(s) Q^*_{\text{MDP}}(s,a). \tag{12.8}$$

Q_{MDP} can be very effective in some domains, but the policies it computes will not take informative actions, as the Q_{MDP} solution assumes that any uncertainty regarding the state will disappear after taking one action. As such, Q_{MDP} policies will fail in domains where repeated information gathering is necessary.

For instance, consider the toy domain in Figure 12.5, which illustrates how MDP-based heuristics can fail (Parr and Russell, 1995). The agent starts in the state marked I, and upon taking any action the system transitions with equal probability to one of two states. In both states it would receive observation A, meaning the agent cannot distinguish between them. The optimal POMDP policy is to take the action a twice in succession, after which the agent is back in the same state. However, because it observed either C or D, it knows in which of the two states marked A it currently is. This knowledge is important for choosing the optimal action (b or c) to transition to the state with positive reward, labelled $+1$. The fact that the a actions do not change the system state, but only the agent's belief state (two time steps later) is very hard for the MDP-based methods to plan for. It forms an example of reasoning about explicit information gathering effects of actions, for which methods based on MDP solutions do not suffice.

One can also expand the MDP setting to model some form of sensing uncertainty without considering full-blown POMDP beliefs. For instance, in robotics the navigation under localization uncertainty problem can be modeled by the mean and entropy of the belief distribution (Cassandra et al, 1996; Roy and Thrun, 2000).

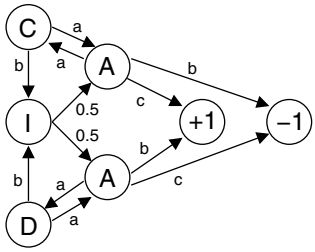


Fig. 12.5 A simple domain in which MDP-based control strategies fail (Parr and Russell, 1995)

Although attractive from a computational perspective, such approaches are likely to fail when the belief is not uni-modal but has a more complex shape.

12.3.2 Value Iteration for POMDPs

To overcome the limitations of MDP-based heuristic methods, we now consider computing optimal POMDP policies via value iteration. The use of belief states allows one to transform the original discrete-state POMDP into a continuous-state MDP. Recall that we can represent a plan in an MDP by its value function, which for every state estimates the amount of discounted cumulative reward the agent can gather when it acts according to the particular plan. In a POMDP the optimal value function, i.e., the value function corresponding to an optimal plan, exhibits particular structure (it is piecewise linear and convex) that one can exploit in order to facilitate computing the solution. Value iteration, for instance, is a method for solving POMDPs that builds a sequence of value-function estimates which converge to the optimal value function for the current task (Sondik, 1971). A value function in a finite-horizon POMDP is parameterized by a finite number of hyperplanes, or vectors, over the belief space, which partition the belief space into a finite amount of regions. Each vector maximizes the value function in a certain region and has an action associated with it, which is the optimal action to take for beliefs in its region.

As we explain next, computing the next value-function estimate—looking one step deeper into the future—requires taking into account all possible actions the agent can take and all subsequent observations it may receive. Unfortunately, this leads to an exponential growth of vectors as the planning horizon increases. Many of the computed vectors will be useless in the sense that their maximizing region is empty, but identifying and subsequently pruning them is an expensive operation.

Exact value-iteration algorithms (Sondik, 1971; Cheng, 1988; Cassandra et al, 1994) search in each value-iteration step the complete belief simplex for a minimal set of belief points that generate the necessary set of vectors for the next-horizon value function. This typically requires linear programming and is therefore costly in high dimensions. Other exact value-iteration algorithms focus on generating all possible next-horizon vectors followed by or interleaved with pruning dominated

vectors in a smart way (Monahan, 1982; Zhang and Liu, 1996; Littman, 1996; Cassandra et al, 1997; Feng and Zilberstein, 2004; Lin et al, 2004; Varakantham et al, 2005). However, pruning again requires linear programming.

The value of an optimal policy π^* is defined by the optimal value function V^* which we compute by iterating a number of stages, at each stage considering a step further into the future. At each stage we apply the exact dynamic-programming operator H_{POMDP} (12.6). If the agent has only one time step left to act, we only have to consider the immediate reward for the particular belief b, and can ignore any future value $V^*(b^{ao})$ and (12.6) reduces to:

$$V_0^*(b) = \max_a \left[\sum_s R(s,a)b(s) \right]. \tag{12.9}$$

We can view the immediate reward function $R(s,a)$ as a set of $|A|$ vectors $\alpha_0^a = (\alpha_0^a(1), \ldots, \alpha_0^a(|S|))$, one for each action a: $\alpha_0^a(s) = R(s,a)$. Now we can rewrite (12.9) as follows, where we view b as a $|S|$-dimensional vector:

$$V_0^*(b) = \max_a \sum_s \alpha_0^a(s)b(s), \tag{12.10}$$

$$= \max_{\{\alpha_0^a\}_a} b \cdot \alpha_0^a, \tag{12.11}$$

where $(\cdot)$ denotes inner product.

In the general case, for $h > 0$, we parameterize a value function V_n at stage n by a finite set of vectors or hyperplanes $\{\alpha_n^k\}$, $k = 1, \ldots, |V_n|$. Given a set of vectors $\{\alpha_n^k\}_{k=1}^{|V_n|}$ at stage n, the value of a belief b is given by

$$V_n(b) = \max_{\{\alpha_n^k\}_k} b \cdot \alpha_n^k. \tag{12.12}$$

Additionally, an action $a(\alpha_n^k) \in A$ is associated with each vector, which is the optimal one to take in the current step, for those beliefs for which α_n^k is the maximizing vector. Each vector defines a region in the belief space for which this vector is the maximizing element of V_n. These regions form a partition of the belief space, induced by the piecewise linearity of the value function, as illustrated by Fig. 12.6.

The gradient of the value function at b is given by the vector

$$\alpha_n^b = \arg\max_{\{\alpha_n^k\}_k} b \cdot \alpha_n^k, \tag{12.13}$$

and the policy at b is given by

$$\pi(b) = a(\alpha_n^b). \tag{12.14}$$

The main idea behind many value-iteration algorithms for POMDPs is that for a given value function V_n and a particular belief point b we can easily compute the vector α_{n+1}^b of $H_{\text{POMDP}}V_n$ such that

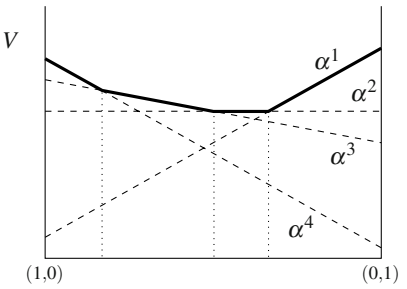


Fig. 12.6 Detailed example of a POMDP value function, c.f. Fig. 12.4a. The value function is indicated by the solid black line, and in this case consists of four α vectors, indicated by dashed lines. The induced partitioning of the belief space into four regions is indicated by the vertical dotted lines.

$$\alpha^b_{n+1} = \underset{\{\alpha^k_{n+1}\}_k}{\arg\max}\, b \cdot \alpha^k_{n+1}, \tag{12.15}$$

where $\{\alpha^k_{n+1}\}_{k=1}^{|H_{\text{POMDP}}V_n|}$ is the (unknown) set of vectors for $H_{\text{POMDP}}V_n$. We will denote this operation $\alpha^b_{n+1} = \texttt{backup}(b)$. For this, we define g_{ao} vectors

$$g^k_{ao}(s) = \sum_{s'} p(o|s',a)p(s'|s,a)\alpha^k_n(s'), \tag{12.16}$$

which represent the vectors resulting from back-projecting α^k_n for a particular a and o. Starting from (12.6) we can derive

$$V_{n+1}(b) = \max_a \left[b \cdot \alpha^a_0 + \gamma b \cdot \sum_o \underset{\{g^k_{ao}\}_k}{\arg\max}\, b \cdot g^k_{ao} \right] \tag{12.17}$$

$$= \max_{\{g^b_a\}_a} b \cdot g^b_a, \tag{12.18}$$

$$\text{with } g^b_a = \alpha^a_0 + \gamma \sum_o \underset{\{g^k_{ao}\}_k}{\arg\max}\, b \cdot g^k_{ao}, \tag{12.19}$$

which can be re-written as

$$V_{n+1}(b) = b \cdot \underset{\{g^b_a\}_a}{\arg\max}\, b \cdot g^b_a. \tag{12.20}$$

From (12.20) we can derive the vector $\texttt{backup}(b)$, as this is the vector whose inner product with b yields $V_{n+1}(b)$:

$$\texttt{backup}(b) = \underset{\{g^b_a\}_{a\in A}}{\arg\max}\, b \cdot g^b_a, \tag{12.21}$$

with g^b_a defined in (12.19). Note that in general not only the computed α vector is retained, but also which action a was the maximizer in (12.21), as that is the optimal action associated with $\texttt{backup}(b)$.

12.3.3 Exact Value Iteration

The Bellman backup operator (12.21) computes a next-horizon vector for a single belief, and now we will employ this backup operator to compute a complete value function for the next horizon, i.e., one that is optimal for all beliefs in the belief space. Although computing the vector $\texttt{backup}(b)$ for a given b is straightforward, locating the (minimal) set of points b required to compute *all* vectors $\cup_b \texttt{backup}(b)$ of $H_{\text{POMDP}}V_n$ is very costly. As each b has a region in the belief space in which its α_n^b is maximal, a family of algorithms tries to identify these regions (Sondik, 1971; Cheng, 1988; Kaelbling et al, 1998). The corresponding b of each region is called a "witness" point, as it testifies to the existence of its region. Other exact POMDP value-iteration algorithms do not focus on searching in the belief space. Instead, they consider enumerating all possible vectors of $H_{\text{POMDP}}V_n$, followed by pruning useless vectors (Monahan, 1982; Zhang and Liu, 1996; Littman, 1996; Cassandra et al, 1997; Feng and Zilberstein, 2004; Lin et al, 2004; Varakantham et al, 2005). We will focus on the enumeration algorithms as they have seen more recent developments and are more commonly used.

12.3.3.1 Monahan's Enumeration Algorithm

First, we consider the most straightforward way of computing $H_{\text{POMDP}}V_n$, due to Monahan (1982). It involves calculating all possible ways $H_{\text{POMDP}}V_n$ could be constructed, exploiting the known structure of the value function. Note that in each $H_{\text{POMDP}}V_n$ a finite number of vectors are generated, as we have assumed finite sets A and O. We operate independently of a particular b now so (12.19) and hence (12.21) can no longer be applied. Instead of maximizing for all $o \in O$ over the g_{ao}^k vectors for the particular b, we now have to include all ways of selecting g_{ao}^k for all o:

$$H_{\text{POMDP}}V_n = \bigcup_a G_a, \text{ with } G_a = \bigoplus_o G_a^o, \text{ and } G_a^o = \left\{ \frac{1}{|O|}\alpha_0^a + \gamma g_{ao}^k \right\}_k, \quad (12.22)$$

where $\bigoplus$ denotes the cross-sum operator.[3]

Unfortunately, at each stage a finite but exponential number of vectors are generated: $|A||V_n|^{|O|}$. The regions of many of the generated vectors will be empty and these vectors are useless as they will not influence the agent's policy. Technically, they are not part of the value function, and keeping them has no effect on subsequent value functions, apart from the computational burden. Therefore, all value-iteration methods in the enumeration family employ some form of pruning. In particular, Monahan (1982) prunes $H_{\text{POMDP}}V_n$ after computing it:

$$V_{n+1} = \texttt{prune}(H_{\text{POMDP}}V_n), \quad (12.23)$$

[3] Cross sum of sets is defined as: $\bigoplus_k R_k = R_1 \oplus R_2 \oplus \cdots \oplus R_k$, with $P \oplus Q = \{\ p+q \mid p \in P,\ q \in Q\ \}$.

with $H_{\text{POMDP}}V_n$ as defined in (12.22). The `prune` operator is implemented by solving a linear program (White, 1991).

12.3.3.2 Incremental Pruning

Monahan (1982)'s algorithm first generates all $|A||V_n|^{|O|}$ vectors of $H_{\text{POMDP}}V_n$ before pruning all dominated vectors. Incremental Pruning methods (Zhang and Liu, 1996; Cassandra et al, 1997; Feng and Zilberstein, 2004; Lin et al, 2004; Varakantham et al, 2005) save computation time by exploiting the fact that

$$\texttt{prune}(G \oplus G' \oplus G'') = \texttt{prune}(\texttt{prune}(G \oplus G') \oplus G''). \tag{12.24}$$

In this way the number of constraints in the linear program used for pruning grows slowly (Cassandra et al, 1997), leading to better performance. The basic Incremental Pruning algorithm exploits (12.24) when computing V_{n+1} as follows:

$$V_{n+1} = \texttt{prune}\big(\bigcup_a G_a\big), \quad \text{with} \tag{12.25}$$

$$G_a = \texttt{prune}\big(\bigoplus_o G_a^o\big) \tag{12.26}$$

$$= \texttt{prune}(G_a^1 \oplus G_a^2 \oplus G_a^3 \oplus \cdots \oplus G_a^{|O|}) \tag{12.27}$$

$$= \texttt{prune}(\cdots \texttt{prune}(\texttt{prune}(G_a^1 \oplus G_a^2) \oplus G_a^3) \cdots \oplus G_a^{|O|}). \tag{12.28}$$

In general, however, computing exact solutions for POMDPs is an intractable problem (Papadimitriou and Tsitsiklis, 1987; Madani et al, 2003), calling for approximate solution techniques (Lovejoy, 1991; Hauskrecht, 2000). Next we present a family of popular approximate value iteration algorithms.

12.3.4 Point-Based Value Iteration Methods

Given the high computational complexity of optimal POMDP solutions, many methods for approximate solutions have been developed. One powerful idea has been to compute solutions only for those parts of the belief simplex that are reachable, i.e., that can be actually encountered by interacting with the environment. This has motivated the use of approximate solution techniques which focus on the use of a sampled set of *belief points* on which planning is performed (Hauskrecht, 2000; Poon, 2001; Roy and Gordon, 2003; Pineau et al, 2003; Smith and Simmons, 2004; Spaan and Vlassis, 2005a; Shani et al, 2007; Kurniawati et al, 2008), a possibility already mentioned by Lovejoy (1991). The idea is that instead of planning over the complete belief space of the agent (which is intractable for large state spaces), planning is carried out only on a limited set of prototype beliefs B that have been sampled by letting the agent interact with the environment.

As we described before, a major cause of intractability of exact POMDP solution methods is their aim of computing the optimal action for every possible belief point in the belief space $\Delta(S)$. For instance, if we use Monahan's algorithm (12.22) we can end up with a series of value functions whose size grows exponentially in the planning horizon. A natural way to sidestep this intractability is to settle for computing an approximate solution by considering only a finite set of belief points. The backup stage reduces to applying (12.21) a fixed number of times, resulting in a small number of vectors (bounded by the size of the belief set). The motivation for using approximate methods is their ability to compute successful policies for much larger problems, which compensates for the loss of optimality.

The general assumption underlying these so-called *point-based* methods is that by updating not only the value but also its gradient (the α vector) at each $b \in B$, the resulting policy will generalize well and be effective for beliefs outside the set B. Whether or not this assumption is realistic depends on the POMDP's structure and the contents of B, but the intuition is that in many problems the set of 'reachable' beliefs (reachable by following an arbitrary policy starting from the initial belief) forms a low-dimensional manifold in the belief simplex, and thus can be covered densely enough by a relatively small number of belief points.

The basic point-based POMDP update operates as follows. It uses an approximate backup operator $\tilde{H}_{\text{PBVI}}$ instead of H_{POMDP}, that in each value-backup stage computes the set

$$\tilde{H}_{\text{PBVI}} V_n = \bigcup_{b \in B} \texttt{backup}(b), \tag{12.29}$$

using a fixed set of belief points B. An alternative randomized backup operator $\tilde{H}_{\text{PERSEUS}}$ is provided by PERSEUS (Spaan and Vlassis, 2005a), which increases (or at least does not decrease) the value of all belief points in B. The key idea is that in each value-backup stage the value of all points in the belief set B can be improved by only backing up a (randomly selected) subset $\tilde{B}$ of the points:

$$\tilde{H}_{\text{PERSEUS}} V_n = \bigcup_{b \in \tilde{B}} \texttt{backup}(b), \tag{12.30}$$

$$\text{ensuring that } V_n(b') \leq V_{n+1}(b'), \forall b' \in B. \tag{12.31}$$

In each backup stage the set $\tilde{B}$ is constructed by sampling beliefs from B until the resulting V_{n+1} upper bounds V_n over B, i.e., until condition (12.31) has been met. The $\tilde{H}_{\text{PERSEUS}}$ operator results in value functions with a relatively small number of vectors, allowing for the use of much larger B, which has a positive effect on the approximation accuracy (Pineau et al, 2003).

Crucial to the control quality of the computed approximate solution is the makeup of B. A number of schemes to build B have been proposed. For instance, one could use a regular grid on the belief simplex, computed, e.g., by Freudenthal triangulation (Lovejoy, 1991). Other options include taking all extreme points of the belief simplex or use a random grid (Hauskrecht, 2000; Poon, 2001). An alternative scheme is to include belief points that can be encountered by simulating the

POMDP: we can generate trajectories through the belief space by sampling random actions and observations at each time step (Lovejoy, 1991; Hauskrecht, 2000; Poon, 2001; Pineau et al, 2003; Spaan and Vlassis, 2005a). This sampling scheme focuses the contents of B to be beliefs that can actually be encountered while experiencing the POMDP model.

More intricate schemes for belief sampling have also been proposed. For instance, one can use the MDP solution to guide the belief sampling process (Shani et al, 2007), but in problem domains which require series of information-gathering actions such a heuristic will suffer from similar issues as when using Q_{MDP} (Section 12.3.1). Furthermore, the belief set B does not need to be static, and can be updated while running a point-based solver. HSVI heuristically selects belief points in the search tree starting from the initial belief, based on upper and lower bounds on the optimal value function (Smith and Simmons, 2004, 2005). SARSOP takes this idea a step further by successively approximating the optimal reachable belief space, i.e., the belief space that can be reached by following an optimal policy (Kurniawati et al, 2008).

In general, point-based methods compute solutions in the form of piecewise linear and convex value functions, and given a particular belief, the agent can simply look up which action to take using (12.14).

12.3.5 Other Approximate Methods

Besides the point-based methods, other types of approximation structure have been explored as well.

12.3.5.1 Grid-Based Approximations

One way to sidestep the intractability of exact POMDP value iteration is to grid the belief simplex, using either a fixed grid (Drake, 1962; Lovejoy, 1991; Bonet, 2002) or a variable grid (Brafman, 1997; Zhou and Hansen, 2001). Value backups are performed for every grid point, but only the value of each grid point is preserved and the gradient is ignored. The value of non-grid points is defined by an interpolation rule. The grid based methods differ mainly on how the grid points are selected and what shape the interpolation function takes. In general, regular grids do not scale well in problems with high dimensionality and non-regular grids suffer from expensive interpolation routines.

12.3.5.2 Policy Search

An alternative to computing an (approximate) value function is policy search: these methods search for a good policy within a restricted class of controllers (Platzman, 1981). For instance, policy iteration (Hansen, 1998b) and bounded policy iteration

(BPI) (Poupart and Boutilier, 2004) search through the space of (bounded-size) stochastic finite-state controllers by performing policy-iteration steps. Other options for searching the policy space include gradient ascent (Meuleau et al, 1999a; Kearns et al, 2000; Ng and Jordan, 2000; Baxter and Bartlett, 2001; Aberdeen and Baxter, 2002) and heuristic methods like stochastic local search (Braziunas and Boutilier, 2004). In particular, the PEGASUS method (Ng and Jordan, 2000) estimates the value of a policy by simulating a (bounded) number of trajectories from the POMDP using a fixed random seed, and then takes steps in the policy space in order to maximize this value. Policy search methods have demonstrated success in several cases, but searching in the policy space can often be difficult and prone to local optima (Baxter et al, 2001).

12.3.5.3 Heuristic Search

Another approach for solving POMDPs is based on heuristic search (Satia and Lave, 1973; Hansen, 1998a; Smith and Simmons, 2004). Defining an initial belief b_0 as the root node, these methods build a tree that branches over (a,o) pairs, each of which recursively induces a new belief node. Branch-and-bound techniques are used to maintain upper and lower bounds to the expected return at fringe nodes in the search tree. Hansen (1998a) proposes a policy-iteration method that represents a policy as a finite-state controller, and which uses the belief tree to focus the search on areas of the belief space where the controller can most likely be improved. However, its applicability to large problems is limited by its use of full dynamic-programming updates. As mentioned before, HSVI (Smith and Simmons, 2004, 2005) is an approximate value-iteration technique that performs a heuristic search through the belief space for beliefs at which to update the bounds, similar to work by Satia and Lave (1973).

12.4 Decision Making Without a-Priori Models

When no models of the environment are available to the agent a priori, the model-based methods presented in the previous section cannot be directly applied. Even relatively simple techniques such as Q_{MDP} (Section 12.3.1) require knowledge of the complete POMDP model: the solution to the underlying MDP is computed using the transition and reward model, while the belief update (12.2) additionally requires the observation model.

In general, there exist two ways of tackling such a decision-making problem, known as direct and indirect reinforcement learning methods. Direct methods apply true model-free techniques, which do not try to reconstruct the unknown POMDP models, but for instance map observation histories directly to actions. On the other extreme, one can attempt to reconstruct the POMDP model by interacting with it, which then in principle can be solved using techniques presented in Section 12.3.

This indirect approach has long been out of favor for POMDPs, as (i) reconstructing (an approximation of) the POMDP models is very hard, and (ii) even with a recovered POMDP, model-based methods would take prohibitively long to compute a good policy. However, advances in model-based methods such as the point-based family of algorithms (Section 12.3.4) have made these types of approaches more attractive.

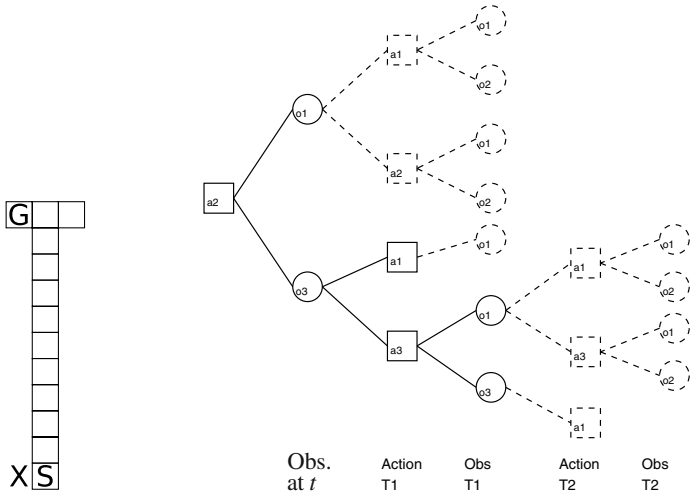
12.4.1 Memoryless Techniques

First, we consider methods for learning memoryless policies, that is, policies that map each observation that an agent receives directly to an action, without consulting any internal state. Memoryless policies can either be deterministic mappings, $\pi : \Omega \to A$, or probabilistic mappings, $\pi : \Omega \to \Delta(A)$. As illustrated by the example in Section 12.2.3, probabilistic policies allow for higher payoffs, at the cost of an increased search space that no longer can be enumerated (Singh et al, 1994). In fact, the problem of finding an optimal deterministic memoryless policy has been shown to be NP-hard (Littman, 1994), while the complexity of determining the optimal probabilistic memoryless policy is still an open problem.

Loch and Singh (1998) have demonstrated empirically that using eligibility traces, in their case in SARSA(λ), can improve the ability of memoryless methods to handle partial observability. SARSA(λ) was shown to learn the optimal deterministic memoryless policy in several domains (for which it was possible to enumerate all such policies, of which there are $|A|^{|\Omega|}$). Bagnell et al (2004) also consider the memoryless deterministic case, but using non-stationary policies instead of stationary ones. They show that successful non-stationary policies can be found in certain maze domains for which no good stationary policies exist. Regarding learning stochastic memoryless policies, an algorithm has been proposed by Jaakkola et al (1995), and tested empirically by Williams and Singh (1999), showing that it can successfully learn stochastic memoryless policies. An interesting twist is provided by Hierarchical Q-Learning (Wiering and Schmidhuber, 1997), which aims to learn a subgoal sequence in a POMDP, where each subgoal can be successfully achieved using a memoryless policy.

12.4.2 Learning Internal Memory

Given the limitations of memoryless policies in systems without a Markovian state signal such as POMDPs, a natural evolution in research has been to incorporate some form of memory, so-called internal state, in each agent. Storing the complete history of the process, i.e., the vector of actions taken by the agent and observations received, is not a practical option for several reasons. First of all, as in the model-free case the agent is not able to compute a belief state, this representation grows without bounds. A second reason is that such a representation does not allow



(a) T Maze. (b) USM suffix tree.

Fig. 12.7 (a) Long-term dependency T maze (Bakker, 2002). (b) Example of a suffix tree used by the USM algorithm (McCallum, 1995), where fringe nodes are indicated by dashed lines.

for easy generalization, e.g., it is not clear how experience obtained after history $\langle a^1, o^1, a^1, o^1 \rangle$ can be used to update the value for history $\langle a^2, o^1, a^1, o^1 \rangle$. To counter these problems, researchers have proposed many different internal-state representations, of which we give a brief overview.

First of all, the memoryless methods presented before can be seen as maintaining a history window of only a single observation. Instead, these algorithms can also be applied with a history window containing the last k observations (Littman, 1994; Loch and Singh, 1998), where k is typically an a-priori defined parameter. In some domains such a relatively cheap increase of the policy space (by means of a low k) can buy a significant improvement in learning time and task performance. Finite history windows have also been used as a representation for neural networks (Lin and Mitchell, 1992).

Finite history windows cannot however capture arbitrary long-term dependencies, such as for instance present in the T Maze in Figure 12.7a, an example provided by Bakker (2002). In this problem the agent starts at S, and needs to navigate to G. However, the location of G is unknown initially, and might be on the left or on the right at the end of the corridor. However, in the start state the agent can observe a road sign X, which depends on the particular goal location. The length of the corridor can be varied (in Figure 12.7a it is 10), meaning that the agent needs to learn to remember the road sign many time steps. Obviously, such a dependency cannot be represented well by finite history windows.

Alleviating the problem of fixed history windows, McCallum (1993, 1995, 1996) proposed several algorithms for variable history windows, among other contributions. These techniques allow for the history window to have a different depth in different parts of the state space. For instance, Utile Suffix Memory (USM) learns a short-term memory representation by growing a suffix tree (McCallum, 1995), an example of which is shown in Figure 12.7b. USM groups together RL experiences based on how much history it considers significant for each instance. In this sense, in different parts of the state space different history lengths can be maintained, in contrast to the finite history window approaches. A suffix tree representation is depicted by solid lines in Figure 12.7b, where the leaves cluster instances that have a matching history up to the corresponding depth. The dashed nodes are the so-called fringe nodes: additional branches in the tree that the algorithm can consider to add to the tree. When a statistical test indicates that instances in a branch of fringe nodes come from different distributions of the expected future discounted reward, the tree is grown to include this fringe branch. Put otherwise, if adding the branch will help predicting the future rewards, it is worthwhile to extend the memory in the corresponding part of the state space. More recent work building on these ideas focuses on better learning behavior in the presence of noisy observations (Shani and Brafman, 2005; Wierstra and Wiering, 2004). Along these lines, recurrent neural networks, for instance based on the Long Short-Term Memory architecture, have also been successfully used as internal state representation (Hochreiter and Schmidhuber, 1997; Bakker, 2002).

Other representations have been proposed as well. Meuleau et al (1999b) extend the VAPS algorithm (Baird and Moore, 1999) to learn policies represented as Finite State Automata (FSA). The FSA represent finite policy graphs, in which nodes are labelled with actions, and the arcs with observations. As in VAPS, stochastic gradient ascent is used to converge to a locally optimal controller. The problem of finding the optimal policy graph of a given size has also been studied (Meuleau et al, 1999a). However, note that the optimal POMDP policy can require an infinite policy graph to be properly represented.

Finally, predictive state representations (PSRs) have been proposed as an alternative to POMDPs for modeling stochastic and partially observable environments (Littman et al, 2002; Singh et al, 2004), see Chapter 13. A PSR dispenses with the hidden POMDP states, and only considers sequences of action and observations which are observed quantities. In a PSR, the state of the system is expressed in possible future event sequences, or "core tests", of alternating actions and observations. The state of a PSR is defined as a vector of probabilities that each core test can actually be realized, given the current history. The advantages of PSRs are most apparent in model-free learning settings, as the model only considers observable events instead of hidden states.

12.5 Recent Trends

To conclude, we discuss some types of approaches that have been gaining popularity recently.

Most of the model-based methods discussed in this chapter are offline techniques that determine a priori what action to take in each situation the agent might encounter. Online approaches, on the other hand, only compute what action to take at the current moment (Ross et al, 2008b). Focusing exclusively on the current decision can provide significant computational savings in certain domains, as the agent does not have to plan for areas of the state space which it never encounters. However, the need to choose actions every time step implies severe constraints on the online search time. Offline point-based methods can be used to compute a rough value function, serving as the online search heuristic. In a similar manner, Monte Carlo approaches are also appealing for large POMDPs, as they only require a generative model (black box simulator) to be available and they have the potential to mitigate the curse of dimensionality (Thrun, 2000; Kearns et al, 2000; Silver and Veness, 2010).

As discussed in detail in Chapter 11, Bayesian RL techniques are promising for POMDPs, as they provide an integrated way of exploring and exploiting models. Put otherwise, they do not require interleaving the model-learning phases (e.g., using Baum-Welch (Koenig and Simmons, 1996) or other methods (Shani et al, 2005)) with model-exploitation phases, which could be a naive approach to apply model-based methods to unknown POMDPs. Poupart and Vlassis (2008) extended the BEETLE algorithm (Poupart et al, 2006), a Bayesian RL method for MDPs, to partially observable settings. As other Bayesian RL methods, the models are represented by Dirichlet distributions, and learning involves updating the Dirichlet hyperparameters. The work is more general than the earlier work by Jaulmes et al (2005), which required the existence of an oracle that the agent could query to reveal the true state. Ross et al (2008a) proposed the Bayes-Adaptive POMDP model, an alternative model for Bayesian reinforcement learning which extends Bayes-Adaptive MDPs (Duff, 2002). All these methods assume that the size of the state, observation and action spaces are known.

Policy gradient methods search in a space of parameterized policies, optimizing the policy by performing gradient ascent in the parameter space (Peters and Bagnell, 2010). As these methods do not require to estimate a belief state (Aberdeen and Baxter, 2002), they have been readily applied in POMDPs, with impressive results (Peters and Schaal, 2008).

Finally, a recent trend has been to cast the model-based RL problem as one of probabilistic inference, for instance using Expectation Maximization for computing optimal policies in MDPs. Vlassis and Toussaint (2009) showed how such methods can also be extended to the model-free POMDP case. In general, inference methods can provide fresh insights in well-known RL algorithms.

Acknowledgements. This work was funded by Fundação para a Ciência e a Tecnologia (ISR/IST pluriannual funding) through the PIDDAC Program funds and was supported by project PTDC/EEA-ACR/73266/2006.

References

Aberdeen, D., Baxter, J.: Scaling internal-state policy-gradient methods for POMDPs. In: International Conference on Machine Learning (2002)

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

Bagnell, J.A., Kakade, S., Ng, A.Y., Schneider, J.: Policy search by dynamic programming. In: Advances in Neural Information Processing Systems, vol. 16. MIT Press (2004)

Baird, L., Moore, A.: Gradient descent for general reinforcement learning. In: Advances in Neural Information Processing Systems, vol. 11. MIT Press (1999)

Bakker, B.: Reinforcement learning with long short-term memory. In: Advances in Neural Information Processing Systems, vol. 14. MIT Press (2002)

Baxter, J., Bartlett, P.L.: Infinite-horizon policy-gradient estimation. Journal of Artificial Intelligence Research 15, 319–350 (2001)

Baxter, J., Bartlett, P.L., Weaver, L.: Experiments with infinite-horizon, policy-gradient estimation. Journal of Artificial Intelligence Research 15, 351–381 (2001)

Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: The complexity of decentralized control of Markov decision processes. Mathematics of Operations Research 27(4), 819–840 (2002)

Bonet, B.: An epsilon-optimal grid-based algorithm for partially observable Markov decision processes. In: International Conference on Machine Learning (2002)

Boutilier, C., Poole, D.: Computing optimal policies for partially observable decision processes using compact representations. In: Proc. of the National Conference on Artificial Intelligence (1996)

Brafman, R.I.: A heuristic variable grid solution method for POMDPs. In: Proc. of the National Conference on Artificial Intelligence (1997)

Braziunas, D., Boutilier, C.: Stochastic local search for POMDP controllers. In: Proc. of the National Conference on Artificial Intelligence (2004)

Brunskill, E., Kaelbling, L., Lozano-Perez, T., Roy, N.: Continuous-state POMDPs with hybrid dynamics. In: Proc. of the Int. Symposium on Artificial Intelligence and Mathematics (2008)

Cassandra, A.R.: Exact and approximate algorithms for partially observable Markov decision processes. PhD thesis, Brown University (1998)

Cassandra, A.R., Kaelbling, L.P., Littman, M.L.: Acting optimally in partially observable stochastic domains. In: Proc. of the National Conference on Artificial Intelligence (1994)

Cassandra, A.R., Kaelbling, L.P., Kurien, J.A.: Acting under uncertainty: Discrete Bayesian models for mobile robot navigation. In: Proc. of International Conference on Intelligent Robots and Systems (1996)

Cassandra, A.R., Littman, M.L., Zhang, N.L.: Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In: Proc. of Uncertainty in Artificial Intelligence (1997)

Cheng, H.T.: Algorithms for partially observable Markov decision processes. PhD thesis, University of British Columbia (1988)

Doshi, F., Roy, N.: The permutable POMDP: fast solutions to POMDPs for preference elicitation. In: Proc. of Int. Conference on Autonomous Agents and Multi Agent Systems (2008)

Drake, A.W.: Observation of a Markov process through a noisy channel. Sc.D. thesis, Massachusetts Institute of Technology (1962)

Duff, M.: Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes. PhD thesis, University of Massachusetts, Amherst (2002)

Dynkin, E.B.: Controlled random sequences. Theory of Probability and its Applications 10(1), 1–14 (1965)

Ellis, J.H., Jiang, M., Corotis, R.: Inspection, maintenance, and repair with partial observability. Journal of Infrastructure Systems 1(2), 92–99 (1995)

Feng, Z., Zilberstein, S.: Region-based incremental pruning for POMDPs. In: Proc. of Uncertainty in Artificial Intelligence (2004)

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

Fox, D., Burgard, W., Thrun, S.: Markov localization for mobile robots in dynamic environments. Journal of Artificial Intelligence Research 11, 391–427 (1999)

Haight, R.G., Polasky, S.: Optimal control of an invasive species with imperfect information about the level of infestation. Resource and Energy Economics (2010) (in Press, Corrected Proof)

Hansen, E.A.: Finite-memory control of partially observable systems. PhD thesis, University of Massachusetts, Amherst (1998a)

Hansen, E.A.: Solving POMDPs by searching in policy space. In: Proc. of Uncertainty in Artificial Intelligence (1998b)

Hansen, E.A., Feng, Z.: Dynamic programming for POMDPs using a factored state representation. In: Int. Conf. on Artificial Intelligence Planning and Scheduling (2000)

Hauskrecht, M.: Value function approximations for partially observable Markov decision processes. Journal of Artificial Intelligence Research 13, 33–95 (2000)

Hauskrecht, M., Fraser, H.: Planning treatment of ischemic heart disease with partially observable Markov decision processes. Artificial Intelligence in Medicine 18, 221–244 (2000)

Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation 9(8), 1735–1780 (1997)

Hoey, J., Little, J.J.: Value-directed human behavior analysis from video using partially observable Markov decision processes. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(7), 1–15 (2007)

Hoey, J., Poupart, P.: Solving POMDPs with continuous or large discrete observation spaces. In: Proc. Int. Joint Conf. on Artificial Intelligence (2005)

Hsiao, K., Kaelbling, L., Lozano-Perez, T.: Grasping pomdps. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 4685–4692 (2007)

Jaakkola, T., Singh, S.P., Jordan, M.I.: Reinforcement learning algorithm for partially observable Markov decision problems. In: Advances in Neural Information Processing Systems, vol. 7 (1995)

Jaulmes, R., Pineau, J., Precup, D.: Active Learning in Partially Observable Markov Decision Processes. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 601–608. Springer, Heidelberg (2005)

Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101, 99–134 (1998)

Kearns, M., Mansour, Y., Ng, A.Y.: Approximate planning in large POMDPs via reusable trajectories. In: Advances in Neural Information Processing Systems, vol. 12. MIT Press (2000)

Koenig, S., Simmons, R.: Unsupervised learning of probabilistic models for robot navigation. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (1996)

Kurniawati, H., Hsu, D., Lee, W.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Robotics: Science and Systems (2008)

Lin, L., Mitchell, T.: Memory approaches to reinforcement learning in non-Markovian domains. Tech. rep., Carnegie Mellon University, Pittsburgh, PA, USA (1992)

Lin, Z.Z., Bean, J.C., White, C.C.: A hybrid genetic/optimization algorithm for finite horizon, partially observed Markov decision processes. INFORMS Journal on Computing 16(1), 27–38 (2004)

Littman, M.L.: Memoryless policies: theoretical limitations and practical results. In: Proc. of the 3rd Int. Conf. on Simulation of Adaptive Behavior: from Animals to Animats 3, pp. 238–245. MIT Press, Cambridge (1994)

Littman, M.L.: Algorithms for sequential decision making. PhD thesis, Brown University (1996)

Littman, M.L., Cassandra, A.R., Kaelbling, L.P.: Learning policies for partially observable environments: Scaling up. In: International Conference on Machine Learning (1995)

Littman, M.L., Sutton, R.S., Singh, S.: Predictive representations of state. In: Advances in Neural Information Processing Systems, vol. 14. MIT Press (2002)

Loch, J., Singh, S.: Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In: International Conference on Machine Learning (1998)

Lovejoy, W.S.: Computationally feasible bounds for partially observed Markov decision processes. Operations Research 39(1), 162–175 (1991)

Madani, O., Hanks, S., Condon, A.: On the undecidability of probabilistic planning and related stochastic optimization problems. Artificial Intelligence 147(1-2), 5–34 (2003)

McCallum, R.A.: Overcoming incomplete perception with utile distinction memory. In: International Conference on Machine Learning (1993)

McCallum, R.A.: Instance-based utile distinctions for reinforcement learning with hidden state. In: International Conference on Machine Learning (1995)

McCallum, R.A.: Reinforcement learning with selective perception and hidden state. PhD thesis, University of Rochester (1996)

Meuleau, N., Kim, K.E., Kaelbling, L.P., Cassandra, A.R.: Solving POMDPs by searching the space of finite policies. In: Proc. of Uncertainty in Artificial Intelligence (1999a)

Meuleau, N., Peshkin, L., Kim, K.E., Kaelbling, L.P.: Learning finite-state controllers for partially observable environments. In: Proc. of Uncertainty in Artificial Intelligence (1999b)

Monahan, G.E.: A survey of partially observable Markov decision processes: theory, models and algorithms. Management Science 28(1) (1982)

Ng, A.Y., Jordan, M.: PEGASUS: A policy search method for large MDPs and POMDPs. In: Proc. of Uncertainty in Artificial Intelligence (2000)

Oliehoek, F.A., Spaan, M.T.J., Vlassis, N.: Optimal and approximate Q-value functions for decentralized POMDPs. Journal of Artificial Intelligence Research 32, 289–353 (2008)

Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of Markov decision processes. Mathematics of Operations Research 12(3), 441–450 (1987)

Parr, R., Russell, S.: Approximating optimal policies for partially observable stochastic domains. In: Proc. Int. Joint Conf. on Artificial Intelligence (1995)

Peters, J., Bagnell, J.A.D.: Policy gradient methods. In: Springer Encyclopedia of Machine Learning. Springer, Heidelberg (2010)
Peters, J., Schaal, S.: Natural actor-critic. Neurocomputing 71, 1180–1190 (2008)
Pineau, J., Thrun, S.: An integrated approach to hierarchy and abstraction for POMDPs. Tech. Rep. CMU-RI-TR-02-21, Robotics Institute, Carnegie Mellon University (2002)
Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for POMDPs. In: Proc. Int. Joint Conf. on Artificial Intelligence (2003)
Platzman, L.K.: A feasible computational approach to infinite-horizon partially-observed Markov decision problems. Tech. Rep. J-81-2, School of Industrial and Systems Engineering, Georgia Institute of Technology, reprinted in working notes AAAI, Fall Symposium on Planning with POMDPs (1981)
Poon, K.M.: A fast heuristic algorithm for decision-theoretic planning. Master's thesis, The Hong-Kong University of Science and Technology (2001)
Porta, J.M., Spaan, M.T.J., Vlassis, N.: Robot planning in partially observable continuous domains. In: Robotics: Science and Systems (2005)
Porta, J.M., Vlassis, N., Spaan, M.T.J., Poupart, P.: Point-based value iteration for continuous POMDPs. Journal of Machine Learning Research 7, 2329–2367 (2006)
Poupart, P.: Exploiting structure to efficiently solve large scale partially observable Markov decision processes. PhD thesis, University of Toronto (2005)
Poupart, P., Boutilier, C.: Bounded finite state controllers. In: Advances in Neural Information Processing Systems, vol. 16. MIT Press (2004)
Poupart, P., Vlassis, N.: Model-based Bayesian reinforcement learning in partially observable domains. In: International Symposium on Artificial Intelligence and Mathematics, ISAIM (2008)
Poupart, P., Vlassis, N., Hoey, J., Regan, K.: An analytic solution to discrete Bayesian reinforcement learning. In: International Conference on Machine Learning (2006)
Ross, S., Chaib-draa, B., Pineau, J.: Bayes-adaptive POMDPs. In: Advances in Neural Information Processing Systems, vol. 20, pp. 1225–1232. MIT Press (2008a)
Ross, S., Pineau, J., Paquet, S., Chaib-draa, B.: Online planning algorithms for POMDPs. Journal of Artificial Intelligence Research 32, 664–704 (2008b)
Roy, N., Gordon, G.: Exponential family PCA for belief compression in POMDPs. In: Advances in Neural Information Processing Systems, vol. 15. MIT Press (2003)
Roy, N., Thrun, S.: Coastal navigation with mobile robots. In: Advances in Neural Information Processing Systems, vol. 12. MIT Press (2000)
Roy, N., Gordon, G., Thrun, S.: Finding approximate POMDP solutions through belief compression. Journal of Artificial Intelligence Research 23, 1–40 (2005)
Sanner, S., Kersting, K.: Symbolic dynamic programming for first-order POMDPs. In: Proc. of the National Conference on Artificial Intelligence (2010)
Satia, J.K., Lave, R.E.: Markovian decision processes with probabilistic observation of states. Management Science 20(1), 1–13 (1973)
Seuken, S., Zilberstein, S.: Formal models and algorithms for decentralized decision making under uncertainty. Autonomous Agents and Multi-Agent Systems (2008)
Shani, G., Brafman, R.I.: Resolving perceptual aliasing in the presence of noisy sensors. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 17, pp. 1249–1256. MIT Press, Cambridge (2005)
Shani, G., Brafman, R.I., Shimony, S.E.: Model-Based Online Learning of POMDPs. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 353–364. Springer, Heidelberg (2005)

Shani, G., Brafman, R.I., Shimony, S.E.: Forward search value iteration for POMDPs. In: Proc. Int. Joint Conf. on Artificial Intelligence (2007)

Shani, G., Poupart, P., Brafman, R.I., Shimony, S.E.: Efficient ADD operations for point-based algorithms. In: Int. Conf. on Automated Planning and Scheduling (2008)

Silver, D., Veness, J.: Monte-carlo planning in large POMDPs. In: Lafferty, J., Williams, C.K.I., Shawe-Taylor, J., Zemel, R., Culotta, A. (eds.) Advances in Neural Information Processing Systems, vol. 23, pp. 2164–2172 (2010)

Simmons, R., Koenig, S.: Probabilistic robot navigation in partially observable environments. In: Proc. Int. Joint Conf. on Artificial Intelligence (1995)

Singh, S., Jaakkola, T., Jordan, M.: Learning without state-estimation in partially observable Markovian decision processes. In: International Conference on Machine Learning (1994)

Singh, S., James, M.R., Rudary, M.R.: Predictive state representations: A new theory for modeling dynamical systems. In: Proc. of Uncertainty in Artificial Intelligence (2004)

Smallwood, R.D., Sondik, E.J.: The optimal control of partially observable Markov decision processes over a finite horizon. Operations Research 21, 1071–1088 (1973)

Smith, T., Simmons, R.: Heuristic search value iteration for POMDPs. In: Proc. of Uncertainty in Artificial Intelligence (2004)

Smith, T., Simmons, R.: Point-based POMDP algorithms: Improved analysis and implementation. In: Proc. of Uncertainty in Artificial Intelligence (2005)

Sondik, E.J.: The optimal control of partially observable Markov processes. PhD thesis, Stanford University (1971)

Spaan, M.T.J., Vlassis, N.: A point-based POMDP algorithm for robot planning. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (2004)

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

Spaan, M.T.J., Vlassis, N.: Planning with continuous actions in partially observable environments. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (2005b)

Spaan, M.T.J., Veiga, T.S., Lima, P.U.: Active cooperative perception in network robot systems using POMDPs. In: Proc. of International Conference on Intelligent Robots and Systems (2010)

Sridharan, M., Wyatt, J., Dearden, R.: Planning to see: A hierarchical approach to planning visual actions on a robot using POMDPs. Artificial Intelligence 174, 704–725 (2010)

Stankiewicz, B., Cassandra, A., McCabe, M., Weathers, W.: Development and evaluation of a Bayesian low-vision navigation aid. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 37(6), 970–983 (2007)

Stratonovich, R.L.: Conditional Markov processes. Theory of Probability and Its Applications 5(2), 156–178 (1960)

Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)

Theocharous, G., Mahadevan, S.: Approximate planning with hierarchical partially observable Markov decision processes for robot navigation. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (2002)

Thrun, S.: Monte Carlo POMDPs. In: Advances in Neural Information Processing Systems, vol. 12. MIT Press (2000)

Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press (2005)

Varakantham, P., Maheswaran, R., Tambe, M.: Exploiting belief bounds: Practical POMDPs for personal assistant agents. In: Proc. of Int. Conference on Autonomous Agents and Multi Agent Systems (2005)

Vlassis, N., Toussaint, M.: Model-free reinforcement learning as mixture learning. In: International Conference on Machine Learning, pp. 1081–1088. ACM (2009)

Wang, C., Khardon, R.: Relational partially observable MDPs. In: Proc. of the National Conference on Artificial Intelligence (2010)
White, C.C.: Partially observed Markov decision processes: a survey. Annals of Operations Research 32 (1991)
Wiering, M., Schmidhuber, J.: HQ-learning. Adaptive Behavior 6(2), 219–246 (1997)
Wierstra, D., Wiering, M.: Utile distinction hidden Markov models. In: International Conference on Machine Learning (2004)
Williams, J.D., Young, S.: Partially observable Markov decision processes for spoken dialog systems. Computer Speech and Language 21(2), 393–422 (2007)
Williams, J.K., Singh, S.: Experimental results on learning stochastic memoryless policies for partially observable Markov decision processes. In: Advances in Neural Information Processing Systems, vol. 11 (1999)
Zhang, N.L., Liu, W.: Planning in stochastic domains: problem characteristics and approximations. Tech. Rep. HKUST-CS96-31, Department of Computer Science, The Hong Kong University of Science and Technology (1996)
Zhou, R., Hansen, E.A.: An improved grid-based approximation algorithm for POMDPs. In: Proc. Int. Joint Conf. on Artificial Intelligence (2001)