POMDP solution methods

Darius Braziunas
Department of Computer Science
University of Toronto

2003

Abstract

This is an overview of partially observable Markov decision processes (POMDPs). We describe POMDP value and policy iteration as well as gradient ascent algorithms. The emphasis is on solution methods that work directly in the space of policies.

Contents

1 Introduction

Partially observable Markov decision processes (POMDPs) provide a natural model for sequential decision making under uncertainty. This model augments a well-researched framework of Markov decision processes (MDPs) [Howard, 1960, Puterman, 1994] to situations where an agent cannot reliably identify the underlying environment state. The POMDP formalism is very general and powerful, extending the application of MDPs to many realistic problems.

Unfortunately, the generality of POMDPs entails high computational cost. The problem of finding optimal policies for finite-horizon POMDPs has been proven to be PSPACE-complete [Papadimitriou and Tsitsiklis, 1987], and their existence for infinite-horizon POMDPs — undecidable [Madani et al., 1999]. Because of the intractability of current solution algorithms, especially those that use dynamic programming to construct (approximately) optimal value functions [Smallwood and Sondik, 1973, Cassandra et al., 1997], the application of POMDPs remains limited to very small problems.

Policy-based solution methods search *directly* in the space of policies for the best course of action. Constraining the policy space facilitates the search and may lead tractable (although approximate) POMDP solution algorithms. *Finite-state controllers* (FSCs) are the policy representation of choice in such work, providing a compromise between the requirement that action choices depend on certain aspects of observable history and the ability to easily control the complexity of policy space being searched.

While optimal FSCs can be constructed if no restrictions are placed on their structure [Hansen, 1998a], it is more usual to impose some structure that one hopes admits a good parameterization, and search through that restricted space. One way is to consider the problem of finding the best FSC *of a given size* for a completely specified POMDP. Even with the FSC size restriction constraint, the problem remains NP-hard [Littman, 1994, Meuleau et al., 1999a]; therefore, *gradient ascent (GA)* has proven to be especially attractive for solving this type of problems because of its computational properties [Meuleau et al., 1999a, Aberdeen and Baxter, 2002]. Unfortunately, gradient-based approaches can converge to arbitrarily bad local optima.

2 Sequential decision processes

A sequential decision process involves an *agent* that interacts synchronously with the external *environment*, or system; the agent's goal is to maximize *reward* by choosing appropriate actions. These actions and the history of the environment *states* determine the probability distribution over possible next states. Therefore, the sequence of system states can be modeled as a stochastic process.

2.1 MDP framework

The most commonly used formal model of fully-observable sequential decision processes is the Markov decision process (MDP) model. An MDP can be viewed as an extension of Markov chains with a set of decisions (actions) and a state-based reward or cost structure. For each possible state of the process, a decision has to be made regarding which action should be executed in that state. The chosen action affects both the transition probabilities and the costs (or rewards) incurred. The goal is to choose an optimal action in every state to increase some predefined measure of performance. The *decision* process for doing this is referred to as the Markov *decision* process.

2.1.1 Actions and state transitions

A state is a description of the environment at a particular point in time. Although we will deal with continuous state and action spaces when describing preference elicitation problems, we generally assume

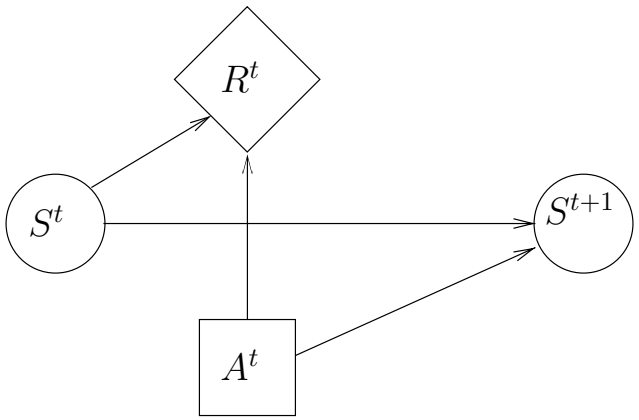

Figure 1: Causal relationships between MDP states, actions, and rewards. R^t is reward received at stage t, i.e., $R(S^t, A^t)$.

that the environment can be in a finite number of states, and the agent can choose from a finite set of actions. Let $\mathcal{S} = \{s_0, s_1, \ldots, s_N\}$ be a finite set of states. Since the process is stochastic, a particular state at some discrete *stage*, or time step $t \in \mathcal{T}$, can be viewed as a random variable S^t whose domain is the state space $\mathcal{S}$.

For a process to be *Markovian*, the state has to contain enough information to predict the next state. This means that the past history of system states is irrelevant to predicting the future:

$$Pr(S^{t+1}|S^0, S^1, \ldots, S^t) = Pr(S^{t+1}|S^t). \tag{1}$$

At each stage, the agent can affect the state transition probabilities by executing one of the available actions. The set of all actions will be denoted by $\mathcal{A}$. Thus, each action $a \in \mathcal{A}$ is fully described by a $|\mathcal{S}| \times |\mathcal{S}|$ *state transition* matrix, whose entry in an ith row and jth column is the probability that the system will move from state s_i to state s_j if action a gets executed:

$$p^a_{ij} = Pr(S^{t+1} = s_j|S^t = s_i, A^t = a). \tag{2}$$

We will assume that our processes are *stationary*, i.e., that the transition probabilities do not depend on the current time step.

The transition function $T(\cdot)$ summarizes the effects of actions on systems states. $T : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ is a function that for each state and action associates a probability distribution over the possible successor states ($\Delta(\mathcal{S})$ denotes the set of all probability distributions over $\mathcal{S}$). Thus, for each $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$, the function T determines the probability of a transition from state s to state s' after executing action a, i.e.,

$$T(s, a, s') = Pr(S^{t+1} = s'|S^t = s,\ A^t = a). \tag{3}$$

2.1.2 Rewards

$R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is a reward function that for each state and action assigns a numeric reward (or cost, if the value is negative). $R(s, a)$ is an immediate reward that an agent would receive for being in state s and executing action a.

The causal relationships between MDP states, actions, and rewards are illustrated in Figure 1.

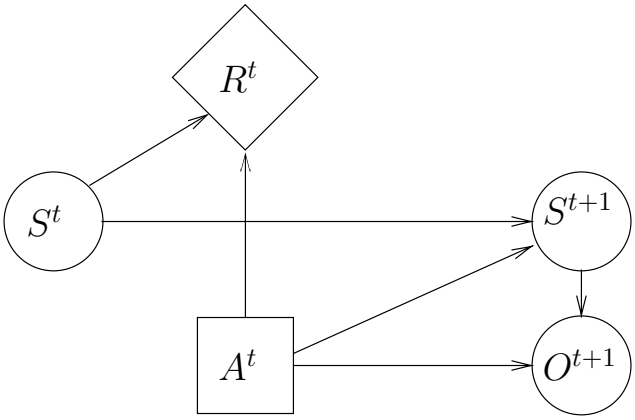

Figure 2: Causal relationships between POMDP states, actions, rewards, and observations.

2.2 POMDP framework

A POMDP is a generalization of MDPs to situations in which system states are not fully observable. This realistic extension of MDPs dramatically increases the complexity of POMDPs, making exact solutions virtually intractable. In order to act optimally, an agent might need to take into account all the previous history of observations and actions, rather than just the current state it is in.

A POMDP is comprised of an underlying MDP, extended with an observation space $\mathcal{O}$ and observation function $Z(\cdot)$.

2.2.1 Observation function

Let $\mathcal{O}$ be a set of observations an agent can receive. In MDPs, the agent has full knowledge of the system state; therefore, $\mathcal{O} \equiv \mathcal{S}$. In partially observable environments, observations are only probabilistically dependent on the underlying environment state. Determining which state the agent is in becomes problematic, because the same observation can be observed in different states.

$Z : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{O})$ is an observation function that specifies the relationship between system states and observations. $Z(s', a, o')$ is the probability that observation o' will be recorded after an agent performs action a and lands in state s':

$$Z(s', a, o') = Pr(O^{t+1} = o' \mid S^{t+1} = s',\ A^t = a). \tag{4}$$

Formally, a POMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, Z \rangle$, consisting of the state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $T(\cdot)$, reward function $R(\cdot)$, observation space $\mathcal{O}$, and observation function $Z(\cdot)$. Its influence diagram is shown in Figure 2.

2.3 Process histories

A *history* is a record of everything that happened during the execution of the process. For POMDPs, a complete system history from the beginning till time t is a sequence of state, observation, and action triples

$$\langle S^0, O^0, A^0 \rangle, \langle S^1, O^1, A^1 \rangle, \ldots, \langle S^t, O^t, A^t \rangle. \tag{5}$$

The set of all complete histories (or *trajectories*) will be denoted as $\mathcal{H}$.

Since rewards depend only on visited states and executed actions, a *system history* is enough to evaluate an agent's performance. Thus, a system history is just a sequence of state and action pairs:

$$\langle S^0, A^0 \rangle, \langle S^1, A^1 \rangle, \ldots, \langle S^t, A^t \rangle. \tag{6}$$

A system history h from the set of all system histories $\mathcal{H}_s$ provides an external, objective view about the process; therefore, value functions will be defined on the set $\mathcal{H}_s$ in the next subsection.

In a partially observable environment, an agent cannot fully observe the underlying world state; therefore, it can only base its decisions on the *observable* history. Let's assume that at the outset, the agent has prior beliefs about the world that are summarized by the probability distribution b_0 over the system states; the agent starts by executing some action a^0 based solely on b_0. The observable history until time step t is then a sequence of action and observation pairs

$$\langle A^0, O^1 \rangle, \langle A^1, O^2 \rangle, \ldots, \langle A^{t-1}, O^t \rangle. \tag{7}$$

The set of all possible observable histories will be denoted as $\mathcal{H}_o$. Different ways of structuring and representing $\mathcal{H}_o$ have resulted in different POMDP solution and policy execution algorithms. The concept of observable history and a closely related notion of internal memory will two central issues discussed in this survey.

2.4 Performance measures

At each step in a sequential decision process, an agent has to decide what action to perform based on its observable history. A policy $\pi : \mathcal{H}_o \mapsto \mathcal{A}$ is a rule that maps observable trajectories into actions. A given policy induces a probability distribution over all possible sequences of states and actions, for an initial distribution b_0. Therefore, an agent has control over the likelihood of particular system trajectories. Its goal is to choose a policy that maximizes some objective function that is defined on the set of system histories $\mathcal{H}_s$.

Such objective function is called a *value* function $V(\cdot)$; it essentially ranks system trajectories by assigning a real number to each $h \in \mathcal{H}_s$; a system history h is preferred to h' if and only if $V(h) > V(h')$. Formally, a value function is a mapping from the set of system histories into real numbers:

$$V : \mathcal{H}_s \mapsto \mathbb{R}. \tag{8}$$

In most MDP and POMDP formulations found in AI literature, the value function $V(\cdot)$ is assumed to have structure that makes it much easier to represent and evaluate. We will make a common assumption that $V(\cdot)$ is *additive* – the value of a particular system history is simply a *sum* of rewards accrued at each time step.

If the decision process stops after a finite number of steps H, the problem is *a finite horizon* problem. In such problems, it is common to maximize the total expected reward. The value function for a system trajectory h of length H is simply the sum of rewards attained at each stage [Bellman, 1957]:

$$V(h) = \sum_{t=0}^{t=H} R(s^t, a^t). \tag{9}$$

The sum of rewards over an infinite trajectory may be unbounded. A mathematically elegant way to address this problem is to introduce a *discount factor* γ; the rewards received later get discounted, and

contribute less than current rewards. The value function for a total discounted reward problem is [Bellman, 1957]:

$$V(h) = \sum_{t=0}^{\infty} \gamma^t \, R(s^t, a^t), \; 0 \leq \gamma < 1. \tag{10}$$

This formulation is very common in current MDP and POMDP literature, including the key papers concerning policy-based search in POMDPs [Hansen, 1997, 1998a, Meuleau et al., 1999a,b]. Another popular value function is the average reward per stage[1], used, e.g., in [Aberdeen and Baxter, 2002].

3 Policy representations

Generally, an agent's task is to calculate the optimal course of action in an uncertain environment and then execute its plan contingent on the history of its sensory inputs. The criterion of optimality is predetermined; here, we will use the infinite horizon discounted sum of rewards model, described above. The agent's behavior is therefore determined by its *policy* π, which in its most general form is a mapping from the set of *observable* histories to actions:

$$\pi : \mathcal{H}_o \mapsto \mathcal{A} \tag{11}$$

Given a history

$$h^t = \langle a^0, o^1 \rangle, \langle a^1, o^2 \rangle, \ldots, \langle a^{t-1}, o^t \rangle,$$

the action prescribed by the policy π at time t would be $a^t = \pi(h^t)$; a^0 is the agent's initial action, and o^t is the latest observation.

One of the more important concepts is that of an *expected policy value*. Taking into account a prior belief distribution over the system states b_0, a policy induces a probability distribution $Pr(h|\pi, b_0)$ over the set of system histories $\mathcal{H}_s$. The expected policy value is simply the expected value of system trajectories induced by the policy π:

$$EV(\pi) \equiv V^\pi = \sum_{h \in \mathcal{H}_s} V(h) Pr(h|\pi, b_0). \tag{12}$$

The value of the policy π at a given starting state s_0 will be denoted $V^\pi(s_0)$. Then,

$$EV(\pi) = \sum_{s \in \mathcal{S}} b_0(s) \, V^\pi(s). \tag{13}$$

The agent's goal is to find a policy $\pi^* \in \Pi$ with the maximal expected value from the set Π of all possible policies.

The general form of a policy as a mapping from arbitrary observation histories to actions is very impractical. Existing POMDP solution algorithms exploit structure in value and observation functions to calculate optimal policies that have much more tractable representations. For example, observable histories can be represented as probability distributions over system states, or grouped into a finite set of distinguishable classes using finite-suffix trees or finite-state controllers.

3.1 MDP policies

A POMDP where an agent can fully observe the underlying system state reduces to an MDP. Since the sequence of states forms a Markov chain, the next state depends only on the current state; the history of the previous states is therefore rendered irrelevant.

[1] $V(h) = \lim_{n \to \infty} \frac{1}{n} \sum_{t=0}^{n} R(s^t, a^t)$.

3.1.1 Finite horizon policies

For finite horizon MDP problems, the knowledge of the current state and stage is sufficient to represent the whole observable trajectory for the purposes of maximizing total reward (discounted or not). Therefore, a policy π can be reduced to a mapping from states and stages to actions:

$$\pi : \mathcal{S} \times \mathcal{T} \mapsto \mathcal{A}. \tag{14}$$

Let $\pi(s,t)$ be the action prescribed by the policy at state s with t stages *remaining* till the end of the process. The expected value of a policy at any state can then be computed by the following recurrence [Bellman, 1957]:

$$\begin{aligned} V_0^\pi(s) &= R(s,\pi(s,0)), \\ V_t^\pi(s) &= R(s,\pi(s,t)) + \gamma \sum_{s' \in \mathcal{S}} T(s,\pi(s,t),s')\ V_{t-1}^\pi(s'). \end{aligned} \tag{15}$$

The value functions in the set $\{V_t^\pi\}_{0 \le t \le H}$ are called *t-horizon*, or *t-step*, value functions; H is the horizon length — a predetermined number of stages the process goes through.

A policy π^* is *optimal* if $V_H^{\pi^*}(s) \ge V_H^{\pi'}(s)$ for all H-horizon policies π' and all states $s \in \mathcal{S}$. The optimal value function is a value function of an optimal policy: $V_H^* \equiv V_H^{\pi^*}$. A key result, called Bellman's *principle of optimality* [Bellman, 1957] allows to calculate the optimal t-step value function from the $(t-1)$-step value function:

$$V_t^*(s) = \max_{a \in \mathcal{A}} \left[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s')\ V_{t-1}^*(s') \right]. \tag{16}$$

This equation has served as a basis for value-iteration MDP solution algorithms and inspired analogous POMDP solution methods.

3.1.2 Infinite horizon policies

For infinite horizon MDP problems, optimal decisions can be calculated based only on the current system state, since at any stage, there is still an infinite number of time steps remaining. Without loss of optimality, infinite horizon policies can be represented as mappings from states to actions [Howard, 1960]:

$$\pi : \mathcal{S} \mapsto \mathcal{A}. \tag{17}$$

Policies that do not depend on stages are called *stationary* policies.

The value of a stationary policy π can be determined by a recurrence analogous to the finite horizon case:

$$V^\pi(s) = R(s,\pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s,\pi(s),s')\ V^\pi(s'). \tag{18}$$

The agent's goal is to find a policy π^* that would maximize the value function $V(\cdot)$ for all states $s \in \mathcal{S}$. The optimal value function is

$$V^*(s) = \max_{a \in \mathcal{A}} \left[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s')\ V^*(s') \right]. \tag{19}$$

3.1.3 Implicit policies

Equations 15 and 18 show how to find the value of a given policy π and provide the basis for policy-iteration algorithms. The calculation is straightforward and amounts to solving a system of linear equations of size $|\mathcal{S}| \times |\mathcal{S}|$.

On the other hand, value-iteration methods employ Equation 16 to calculate optimal value functions directly. Optimal policies can then be defined implicitly by value functions. First, we introduce a notion of a Q-function, or Q-value: $Q(s, a)$ is the value of executing action a at state s, and then following the optimal policy:

$$Q(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s')\ V^*(s'). \tag{20}$$

The optimal infinite horizon policy is a greedy policy with respect to the optimal value function $V^*(\cdot)$:

$$\pi^*(s) = \arg\max_a Q(s, a). \tag{21}$$

3.1.4 Stochastic policies

A stochastic infinite horizon MDP policy is a generalization of a deterministic policy; instead of prescribing a single action to a state, it assigns a *distribution* over all actions to a state. That is, a stochastic policy

$$\psi : \mathcal{S} \mapsto \Delta(\mathcal{A}) \tag{22}$$

maps a state to a probability distribution over actions; $\psi(s, a)$ is the probability that action a will be executed at state s. By incorporating expectation over actions, we can rewrite the Equation 18 for stochastic policies in a straightforward manner:

$$V^\psi(s) = \sum_{a \in \mathcal{A}} \psi(s, a)\ R(s, a) + \gamma \sum_{s',\, a} \psi(s, a)\ T(s, a, s')\ V^\psi(s'). \tag{23}$$

While stochastic policies have no advantage for infinite horizon MDPs, we will use them in solving partially observable MDPs. Making policies stochastic allows to convert the discrete action space into a continuous space of distributions over actions. We can then optimize the value function using continuous optimization techniques.

3.2 POMDP policy trees

In partially observable environments, an agent can only base its decisions on the history of its actions and observations. Instead of a simple mapping from system states to actions, a generic POMDP policy assumes a more complicated form.

As for MDPs, we will first consider finite horizon policies. With one stage left, all an agent can do is to execute an action; with two stages left, it can execute an action, receive an observation, and then execute the final action. For a finite horizon of length H, a policy is a *tree* of height H. Since the number of actions and observations is finite, the set of all policies for horizon H can be represented by a *finite* set of policy trees.

Figure 3 illustrates the concept of a t-horizon policy tree. Each node prescribes an action to be taken at a particular stage; then, an observation received determines the branch to follow. A policy tree for a horizon of length H contains

$$\sum_{t=0}^{t=H-1} |\mathcal{O}|^t = \frac{|\mathcal{O}|^H - 1}{|\mathcal{O}| - 1} \tag{24}$$

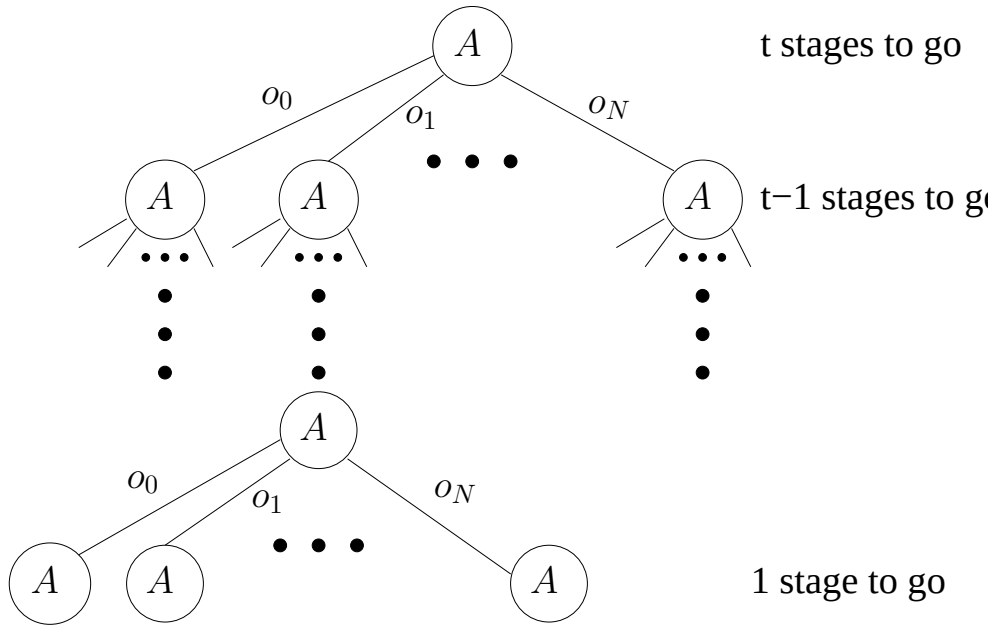

Figure 3: A policy tree for horizon t. For each observation, there is a branch to nodes at a lower level. Each node can be labeled with any action from the set $\mathcal{A}$.

nodes. At each node, there are $|\mathcal{A}|$ choices of actions. Therefore, the size of the set of all possible H-horizon policy trees is

$$|\mathcal{A}|^{\frac{|\mathcal{O}|^H-1}{|\mathcal{O}|-1}}. \tag{25}$$

We will now present a recursive definition of policy trees using an important notion of *conditional plans*. A conditional plan $\sigma \in \Gamma$ is a pair $\langle a, \nu \rangle$ where $a \in \mathcal{A}$ is an action, and $\nu : \mathcal{O} \mapsto \Gamma$ is an *observation strategy*. The set of all observation strategies will be denoted as $\Gamma^{\mathcal{O}}$; obviously, its size is $|\Gamma|^{|\mathcal{O}|}$.

A particular conditional plan tells an agent what action to perform, and what to do next contingent on an observation received. Let Γ_t be the set of all conditional plans available to an agent with t stages left:

$$\Gamma_t = \{\langle a, \nu_t \rangle \mid a \in \mathcal{A},\ \nu_t \in \Gamma_{t-1}^{\mathcal{O}}\}. \tag{26}$$

In this case, $\nu_t : \mathcal{O} \mapsto \Gamma_{t-1}$ is a stage-dependent observation strategy. As a tree of height t can be defined recursively in terms of its subtrees of height $t-1$, so the conditional plans of horizon t can be defined in terms of conditional plans of horizon $t-1$. At the last time step, a conditional plan simply returns an action. A policy tree therefore directly corresponds to a conditional plan. We will use the set Γ_t to denote both the set of t-step policy trees and the equivalent set of conditional plans.

Representing policy trees as conditional plans allows us to write down a recursive expression for their value function. The value function of a non-stationary policy π_t represented by a t-horizon conditional plan $\sigma_t = \langle a, \nu_t \rangle$ is

$$\begin{aligned} V_0^{\pi}(s) &= R(s, \sigma_0(s)), \\ V_t^{\pi}(s) &= V_t^{\sigma_t}(s) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') \sum_{o \in \mathcal{O}} Z(s',a,o)\, V_{t-1}^{\nu_t(o)}(s'), \end{aligned} \tag{27}$$

where $\sigma_0(s)$ is the action to be executed at the last stage.

Since the actual system state is not fully known, we need to calculate the value of a particular policy tree with respect to a (initial) belief state b. Such value is just an expectation of executing the conditional plan σ_t at each state $s \in \mathcal{S}$:

$$V_t^{\pi}(b) = V_t^{\sigma_t}(b) = \sum_{s \in \mathcal{S}} b(s)\, V_t^{\sigma_t}(s). \tag{28}$$

The optimal t-step value function for the belief state b can be found simply by enumerating all the possible policy trees in the set Γ_t:

$$V_t^*(b) = \max_{\sigma \in \Gamma_t} \sum_{s \in \mathcal{S}} b(s)\, V_t^{\sigma}(s). \tag{29}$$

Thus, the t-step value function for the continuous belief simplex $\mathcal{B}$ can in principle be represented by a finite (although doubly exponential in t!) set of conditional plans and a $\max$ operator. The next section discusses some ways of making such a representation more tractable.

3.3 α-vectors and belief state MDPs

The previous Equation 29 actually illustrates the fact that the optimal t-step POMDP value function is piecewise linear and convex [Sondik, 1971, 1978]. From Equation 28 we can see that the value of any policy tree V_t^{σ} is linear in b; hence, from Equation 29, V_t^* is simply the upper surface of the collection of value functions of policies in Γ_t.

Let α^{σ} be a vector of size $|\mathcal{S}|$ whose entries are the values of the conditional plan σ (or, values of a policy tree corresponding to σ) for each state s:

$$\alpha^{\sigma} = [V^{\sigma}(s_0), V^{\sigma}(s_1), \ldots, V^{\sigma}(s_N)]. \tag{30}$$

Equation 29 can then be rewritten in terms of α-vectors :

$$V_t^*(b) = \max_{\sigma \in \Gamma_t} \sum_{s \in \mathcal{S}} b(s)\, \alpha^{\sigma}(s) = \max_{\alpha \in \mathcal{V}_t} \sum_{s \in \mathcal{S}} b(s)\ \alpha(s). \tag{31}$$

Here, the set $\mathcal{V}_t$ contains all t-step α-vectors ; these vectors correspond to t-step policy trees and are sufficient to define the optimal t-horizon value function.

The optimal value function V_t is represented by the upper surface of the α-vectors in $\mathcal{V}_t$ (see Figure 4). Although in the worst case any policy in Γ_t might be superior for some belief region, this rarely happens in practice. Many vectors in the set $\mathcal{V}_t$ might be *dominated* by other vectors, and therefore not needed to represent the optimal value function. In Figure 4, vector α_3 is *pointwise* dominated by α_1, whereas vector α_1 is jointly dominated by the useful vectors α_0 and α_2 together.

Given the set of all α-vectors $\mathcal{V}_t$, it is possible to *prune* it down to a *parsimonious* subset $\mathcal{V}_t^-$ that represents the same optimal value function V_t^*:

$$V_t^*(b) = \max_{\alpha \in \mathcal{V}_t} \sum_{s \in \mathcal{S}} b(s)\ \alpha(s) = \max_{\alpha \in \mathcal{V}_t^-} \sum_{s \in \mathcal{S}} b(s)\ \alpha(s). \tag{32}$$

In a parsimonious set, all α-vectors (or corresponding policy trees) are *useful* [Kaelbling et al., 1998]. A vector α is useful if there is a non-empty belief region $\mathcal{R}(\alpha, \mathcal{V})$ over which it dominates all other vectors, where

$$\mathcal{R}(\alpha, \mathcal{V}) = \{b \mid b \cdot \alpha > b \cdot \alpha',\ \alpha' \in \mathcal{V} - \{\alpha\},\ b \in \mathcal{B}\}. \tag{33}$$

The existence of such region can be easily determined using linear programming. Various value-based POMDP solution algorithms differ in their methods of pruning the set of all α-vectors $\mathcal{V}_t$ to a parsimonious subset $\mathcal{V}_t^-$.

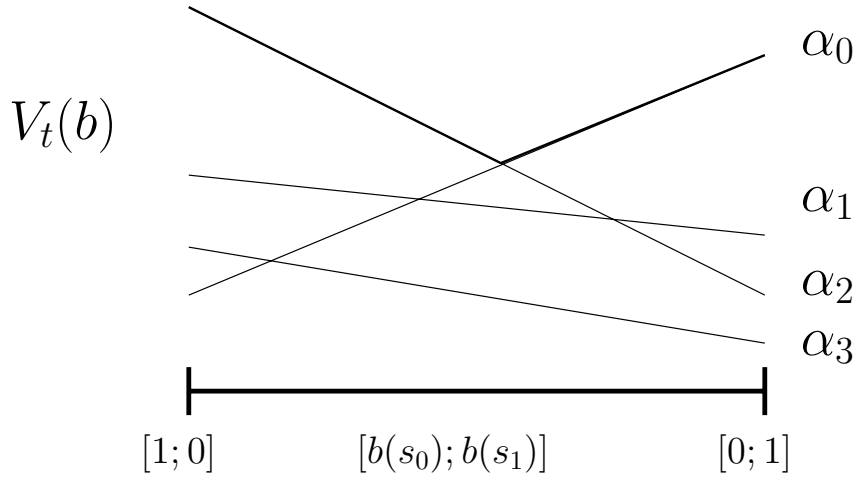

Figure 4: For a two-state POMDP, the belief space $\mathcal{B}$ is a one-dimensional unit interval, since $b(s_0) = Pr(s_0) = 1 - Pr(s_1)$. The horizontal axis therefore represents the whole belief space $\mathcal{B}$ on which the value function $V_t(b)$ is defined. $V_t(b)$ is the upper surface of four α-vectors . Only two of them, α_0 and α_2, are useful.

3.3.1 Implicit POMDP policies

As we already know, an explicit t-step POMDP policy can be represented by a policy tree or a recursive conditional plan. Given an initial belief state b_0, the optimal t-step policy can be found by going through the set of all useful policy trees and finding the one whose value function is maximal with respect to b_0 (see Equation 31). Then, executing the finite horizon policy is straightforward: an agent only needs to perform actions at the nodes, and follow the observation links to policy subtrees.

Instead of keeping all policy trees, it is enough to maintain the set of useful α-vectors $\mathcal{V}_t^-$ for each stage t. As for MDPs, an *implicit* t-step policy can be defined by doing a greedy one-step lookahead. First, we will define the Q-value function $Q_t(b, a)$ as a value of taking action a at belief state b and continuing optimally for the remaining $t - 1$ stages:

$$Q_t(b, a) = \sum_{s \in \mathcal{S}} b(s)R(s, a) + \gamma \sum_{o \in \mathcal{O}} Pr(o|a, b)V^*_{t-1}(b^a_o)), \tag{34}$$

where b^a_o is the belief state that results from b after taking action a and receiving observation o. As we will see below, it can be calculated using the POMDP model and Bayes' theorem.

The optimal action to take at b with t stages remaining is simply

$$\pi^*(b, t) = \arg\max_{a \in \mathcal{A}} Q_t(b, a). \tag{35}$$

3.3.2 Belief state MDPs

A finite horizon POMDP policy now becomes a mapping from belief states and stages to actions:

$$\pi : \mathcal{B} \times \mathcal{T} \mapsto \mathcal{A}. \tag{36}$$

Astrom has shown that a properly updated probability distribution over the state space $\mathcal{S}$ is sufficient to summarize all the observable history of a POMDP agent without loss of optimality [Aström, 1965].

Therefore, a POMDP can be cast into a framework of a fully observable MDP where belief states comprise the continuous, but fully observable, MDP state space. A belief state MDP is therefore a quadruple $\langle \mathcal{B}, \mathcal{A}, T^b, R^b \rangle$, where

- $\mathcal{B} = \Delta(\mathcal{S})$ is the continuous state space.
- $\mathcal{A}$ is the action space, which is the same as in the original POMDP.
- $T^b : \mathcal{B} \times \mathcal{A} \mapsto \mathcal{B}$ is the belief transition function:

$$\begin{aligned} T^b(b, a, b') &= Pr(b'|b, a) \\ &= \sum_{o \in \mathcal{O}} Pr(b'|a, b, o)\, Pr(o|a, b) \\ &= \sum_{o \in \mathcal{O}} Pr(b'|a, b, o) \sum_{s' \in \mathcal{S}} Z(s', a, o) \sum_{s \in \mathcal{S}} T(s, a, s')\, b(s), \end{aligned} \tag{37}$$

  where

$$Pr(b'|a, b, o) = \begin{cases} 1 & \text{if } b_o^a = b', \\ 0 & \text{otherwise.} \end{cases} \tag{38}$$

  After action a and observation o, the updated belief b_o^a can be calculated from the previous belief b:

$$b_o^a(s') = \frac{Z(s', a, o) \sum_{s \in \mathcal{S}} T(s, a, s')\, b(s)}{Pr(o|a, b)}. \tag{39}$$

- $R^b : \mathcal{B} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function:

$$R^b(b, a) = \sum_{s \in \mathcal{S}} b(s)\, R(s, a). \tag{40}$$

To follow the policy that maps from belief states to actions, the agent simply has to execute the action prescribed by the policy, and then update its probability distribution over the system states according to Equation 39.

The infinite horizon optimal value function remains convex, but not necessarily piecewise linear, although it can be approximated arbitrarily closely by a piecewise linear and convex function [Sondik, 1978]. The optimal policy for infinite horizon problems is then just a stationary mapping from belief space to actions:

$$\pi : \mathcal{B} \mapsto \mathcal{A}. \tag{41}$$

It can be extracted by performing a greedy one-step lookahead with respect to the optimal value function V^*:

$$\begin{aligned} Q(b, a) &= \sum_{s \in \mathcal{S}} b(s) R(s, a) + \gamma \sum_{o \in \mathcal{O}} Pr(o|a, b) V^*(b_o^a), \\ \pi^*(b) &= \arg\max_{a \in \mathcal{A}} Q(b, a). \end{aligned} \tag{42}$$

3.4 Finite-state controllers

The optimal infinite horizon value function V^* can be approximated arbitrarily closely by successive finite horizon value functions $V_0, V_1, \ldots, V_t$, as $t \rightarrow \infty$ [Sondik, 1978]. While all optimal t-horizon policies are piecewise-linear and convex, this is not always true for infinite horizon value functions. They remain convex [White and Harrington, 1980], but may contain infinitely many facets.

Some optimal value functions do remain piecewise linear; therefore, at some horizon t, the two successive value functions V_t and V_{t+1} are equal, and therefore, optimal:

$$V^* = V_t = V_{t+1}. \tag{43}$$

Each vector α in a parsimonious set $\mathcal{V}^*$ that represents the optimal infinite horizon value function V^* has an associated belief space region $\mathcal{R}(\alpha, \mathcal{V}^*)$ over which it dominates all other vectors (see Equation 33):

$$\mathcal{R}(\alpha, \mathcal{V}^*) = \{b \mid b \cdot \alpha > b \cdot \alpha',\ \alpha' \in \mathcal{V}^* - \{\alpha\},\ b \in \mathcal{B}\}.$$

Thus, α-vectors define a partition of the belief space. In addition, it has been shown that for each partition there is an optimal action [Smallwood and Sondik, 1973]. When an optimal value function V^* can be represented by a finite set of vectors, all belief states within one region get transformed to new belief states within the *same* single belief partition, given the optimal action and a resulting observation. The set of partitions and belief transitions constitute a *policy graph*, where nodes correspond to belief space partitions with optimal actions attached, and transitions are guided by observations [Cassandra et al., 1994].

Another way of understanding the concept of policy graphs is illustrated in an article by Kaelbling *et al.* [Kaelbling et al., 1998]. If the finite horizon value functions V_t and V_{t+1} become equal, at every level above t the corresponding conditional plans have the same value. Then, it is possible to redraw the observation links from one level to itself as if it were the succeeding level (see Figure 5). Essentially, we can convert non-stationary t-step policy trees (which are non-cyclic policy graphs) into stationary cyclic policy graphs. Such policy graphs enable an agent to execute policies simply by doing actions prescribed at the nodes, and following observation links to successor nodes. The nodes partition the belief space in a way that, for a given action and observation, all belief states in a particular region map to a single region (represented by another graph node).[2] Therefore, an agent does not have to explicitly maintain its belief state and perform expensive operations of updating its beliefs and finding the best α-vector for the belief state. The starting node is optimized for the initial belief state.

Of course, not all POMDP problems allow for optimal infinite horizon policies to be represented by a finite policy graph. Since such a graph cannot be extracted from a suboptimal value function, a policy in such cases is usually defined implicitly by a value function and calculated using Equation 35.

However, limiting the size of a policy provides a tractable way of solving POMDPs *approximately*. Although generally the optimal policy depends on the whole history of observations and actions, one way of facilitating the solution of POMDPs is to assume that an agent has a finite memory. We can represent this finite memory by a set of internal states $\mathcal{N}$. The internal states are fully observable; therefore an agent can execute a policy that maps from *internal* states to actions.

The *action selection* function determines what action to execute at each internal memory state $n \in \mathcal{N}$. In addition to the mapping from internal states to actions, we also need to specify the dynamics of the internal process, i.e., describe the transitions from one internal state to another. The internal memory states can be viewed as nodes, and the transitions between nodes will depend on observations received. Together, the set of nodes and the transition function constitute a policy graph, or a *finite-state controller* (FSC).

[2] Note that this is true only if the optimal infinite horizon value function can be represented by a finite number of α-vectors.

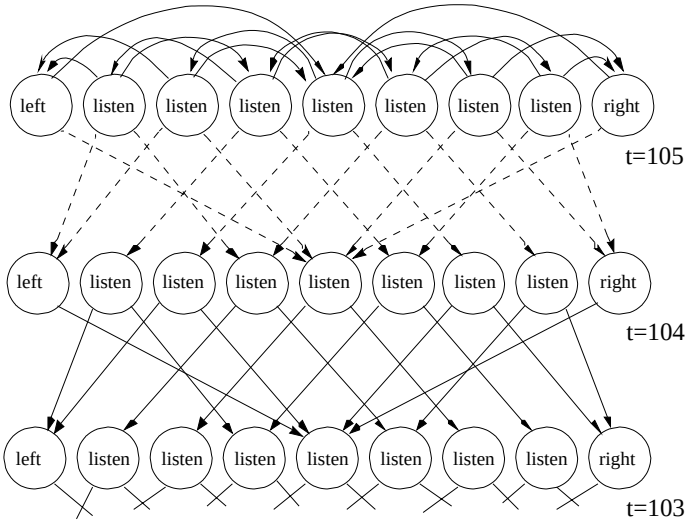

Figure 5: An example from [Kaelbling et al., 1998] that illustrates how policy tree branches can be rearranged to form a stationary policy.

3.4.1 FSC model

A deterministic policy graph π is a triple $\langle \mathcal{N}, \psi, \eta \rangle$, where

- $\mathcal{N}$ is a set of controller nodes n, also known as internal memory states.
- $\psi : \mathcal{N} \mapsto \mathcal{A}$ is the action selection function that for each node n prescribes an action $\psi(n)$.
- $\eta : \mathcal{N} \times \mathcal{O} \mapsto \mathcal{N}$ is the node transition function that for each node and observation assigns a successor node n'. $\eta(n, \cdot)$ is essentially an observation strategy for the node n, described above when discussing policy trees and conditional plans.

In a *stochastic* FSC, the action selection function ψ and the internal transition function η are stochastic. Here,

- $\psi : \mathcal{N} \mapsto \Delta(A)$ is the stochastic action selection function that for each node n prescribes a distribution over actions:
$$\psi(n, a) = Pr(A^t = a | N^t = n). \tag{44}$$
- $\eta : \mathcal{N} \times \mathcal{O} \mapsto \Delta(\mathcal{N})$ is the stochastic node transition function that for each node and observation assigns a probability distribution over successor nodes n'; $\eta(n, o, n')$ is the probability of transition from node n to node n' after observing $o' \in \mathcal{O}$:
$$\eta(n, o', n') = Pr(N^{t+1} = n' | N^t = n,\, O^{t+1} = o'). \tag{45}$$

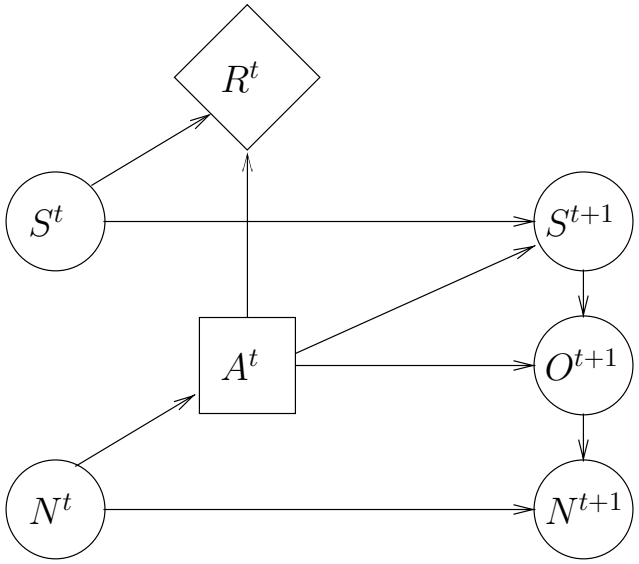

Figure 6: The joint influence diagram for a policy graph and a POMDP. The sequence of FSC nodes coupled with POMDP states is Markovian.

3.5 Cross-product MDP

In the way that an MDP policy $\pi : \mathcal{S} \mapsto \Delta(\mathcal{A})$ gives rise to a Markov chain defined by the transition matrix T, a POMDP policy, represented by a finite graph, is also sufficient to render the dynamics of a POMDP Markovian. The cross-product between the POMDP and the finite policy graph is itself a finite MDP, which will be referred to as the *cross-product MDP*. The structure of both the POMDP and the policy graph can be represented in the cross-product MDP. The influence diagram for such a *coupled* process is shown in Figure 6.

Given a POMDP $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, Z \rangle$ and a policy graph with the node set $\mathcal{N}$, the new cross-product MDP $\langle \bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{T}, \bar{R} \rangle$ can be described as follows [Meuleau et al., 1999a]:

- The state space $\bar{\mathcal{S}} = \mathcal{N} \times \mathcal{S}$ is the Cartesian product of external system states and internal memory nodes; it consists of pairs $\langle n, s \rangle,\ n \in \mathcal{N},\ s \in \mathcal{S}$.
- At each state $\langle n, s \rangle$, there is a choice of action $a \in \mathcal{A}$, and a conditional observation strategy $\nu : \mathcal{O} \mapsto \mathcal{N}$, which determines the next internal node for each possible observation. The new action space $\bar{\mathcal{A}} = \mathcal{A} \times \mathcal{N}^{\mathcal{O}}$ is therefore a cross product between A and the space of observation mappings $\mathcal{N}^{\mathcal{O}}$. A pair $\langle a, \nu \rangle$ is a *conditional plan*, where $a \in \mathcal{A}$ is an action and $\nu \in \mathcal{N}^{\mathcal{O}}$ is a deterministic observation strategy.
- $\bar{T} : \bar{\mathcal{S}} \times \bar{\mathcal{A}} \mapsto \bar{\mathcal{S}}$ is the transition function:

$$\bar{T}(\langle n, s \rangle, \langle a, \nu \rangle, \langle n', s' \rangle) = T(s, a, s') \sum_{o | \nu(o) = n'} Z(s', a, o). \tag{46}$$

- The reward function $\bar{R} : \bar{\mathcal{S}} \times \bar{\mathcal{A}} \mapsto \mathbb{R}$ becomes:

$$\bar{R}(\langle n, s \rangle, \langle a, \nu \rangle) = R(s, a). \tag{47}$$

3.5.1 Policy graph value

Given a (stochastic) policy graph $\pi = \langle \mathcal{N}, \psi, \eta \rangle$ and a POMDP $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, Z \rangle$, the generated sequence of node-state pairs $\langle N^t, S^t \rangle$ constitutes a Markov chain [Hansen, 1997, 1998a, Meuleau et al., 1999a]. In a way analogous to Equation 23, the value of a given policy graph can be calculated using Bellman's equations:

$$\bar{V}^\pi(\bar{s}) = \bar{R}^\pi(\bar{s}) + \gamma \sum_{\bar{s}'} \bar{T}^\pi(\bar{s}, \bar{s}')\, \bar{V}^\pi(\bar{s}'), \tag{48}$$

where $\bar{s}, \bar{s}'$ are node-state pairs in $\bar{\mathcal{S}}$, and

- $\bar{T}^\pi$ is the transition matrix. Given stochastic functions $\psi(\cdot)$ and $\eta(\cdot)$, the transition matrix is analogous to Equation 23 for MDPs, although now we need to take expectation not only over actions a, but also over observations o:

$$\bar{T}^\pi(\langle n, s \rangle, \langle n', s' \rangle) = \sum_{a,o} \psi(n, a)\, \eta(n, o, n')\, T(s, a, s')\, Z(s', a, o). \tag{49}$$

- $\bar{R}^\pi$ is the reward vector:

$$\bar{R}^\pi(\langle n, s \rangle) = \sum_{a} \psi(n, a) R(s, a). \tag{50}$$

4 Exact solution algorithms

4.1 Value iteration

4.1.1 MDP value iteration

Value iteration for MDPs is a standard method of finding the optimal infinite horizon policy π^* using a sequence of optimal finite horizon value functions $V_0^*, V_1^*, \ldots, V_t^*$ [Howard, 1960]. The difference between the optimal value function and the optimal t-horizon value function goes to zero as t goes to infinity:

$$\lim_{t \to \infty} \max_{s \in \mathcal{S}} |V^*(s) - V_t^*(s)| = 0. \tag{51}$$

It turns out that the optimal value function can be calculated in a finite number of steps given the *Bellman error* ϵ, which is the maximum difference (for all states) between two successive finite horizon value functions. Using Equation 16, the value iteration algorithm for MDPs can be summarized as follows:

- Initialize $t = 0$ and $V_0(s) = 0$ for all $s \in \mathcal{S}$.
- While $\max_{s \in \mathcal{S}} |V_{t+1}(s) - V_t(s)| > \epsilon$, calculate $V_{t+1}(s)$ for all states $s \in \mathcal{S}$ according to the following equation, and then increment t:

$$V_{t+1}(s) = \max_{a \in \mathcal{A}} \left[R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s')\, V_t(s') \right].$$

This algorithm results in an implicit policy (which can be extracted using Equation 21) that is within $2\epsilon\gamma/(1-\gamma)$ of the optimal [Bellman, 1957].

4.1.2 POMDP value iteration

As described above, any POMDP can be reduced to a continuous belief-state MDP. Therefore, value iteration can also be used to calculate optimal infinite horizon POMDP policies:

- Initialize $t = 0$ and $V_0(b) = 0$ for all $b \in \mathcal{B}$.
- While $\sup_{b \in \mathcal{B}} |V_{t+1}(b) - V_t(b)| > \epsilon$, calculate $V_{t+1}(b)$ for all states $b \in \mathcal{B}$ according to the following equation, and then increment t:

$$V_{t+1}(b) = \max_{a \in \mathcal{A}} \left[R^b(b,a) + \gamma \sum_{b' \in \mathcal{B}} T^b(b,a,b') \, V_t(b') \right]. \tag{52}$$

The previous equation can be rewritten in terms of the original POMDP formulation as

$$V_{t+1}(b) = \max_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} b(s) R(s,a) + \gamma \sum_{o \in \mathcal{O}} Pr(o|a,b) V_t(b_o^a) \right], \tag{53}$$

where $Pr(o|a,b)$ is

$$Pr(o|a,b) = \sum_{s' \in \mathcal{S}} Z(s',a,o) \sum_{s \in \mathcal{S}} T(s,a,s') \, b(s). \tag{54}$$

Although the belief space is continuous, any optimal finite horizon value function is piecewise linear and convex and can be represented as a finite set of α-vectors (see Section 3.3). Therefore, the essential task of all value-iteration POMDP algorithms is to find the set $\mathcal{V}_{t+1}$ representing value function V_{t+1}, given the previous set of α-vectors $\mathcal{V}_t$.

Various POMDP algorithms differ in how they compute value function representations. The most naive way is to construct the set of conditional plans $\mathcal{V}_{t+1}$ by enumerating all the possible actions and observation mappings to the set $\mathcal{V}_t$. The size of $\mathcal{V}_{t+1}$ is then $|\mathcal{A}||\mathcal{V}_t|^{|\mathcal{O}|}$. Since many vectors in $\mathcal{V}_t$ might be dominated by others, the optimal t-horizon value function can be represented by a parsimonious set $\mathcal{V}_t^-$. The set $\mathcal{V}_t^-$ is the smallest subset of $\mathcal{V}_t$ that still represents the same value function V_t^*; all α-vectors in $\mathcal{V}_t^-$ are useful at some belief state (see Section 3.3). To compute $\mathcal{V}_{t+1}$ (and $\mathcal{V}_{t+1}^-$), we only need to consider the parsimonious set $\mathcal{V}_t^-$.

Some algorithms calculate $\mathcal{V}_{t+1}^-$ by generating $\mathcal{V}_{t+1}$ of size $|\mathcal{A}||\mathcal{V}_t^-|^{|\mathcal{O}|}$, and then *pruning* dominated α-vectors, usually by linear programming. Such algorithms include Monahan's algorithm [Monahan, 1982, **?**], and Incremental pruning [**?**Cassandra et al., 1997]. Other methods, such as Sondik's One-pass [Sondik, 1971, Smallwood and Sondik, 1973], Cheng's Linear Support [Cheng, 1988], and Witness [Kaelbling et al., 1998], build the set $\mathcal{V}_{t+1}^-$ directly from the previous set $\mathcal{V}_t^-$, without considering non-useful conditional plans. Even the fastest of exact value-iteration algorithms can currently solve only toy problems.

As for MDPs, for a given ϵ, the implicit policy extracted from the value function is within $2\epsilon\gamma/(1-\gamma)$ of the optimal policy value.

4.2 Policy iteration

Policy iteration algorithms proceed by iteratively improving the policies themselves. The sequence $\pi_0, \pi_1, \ldots, \pi_t$ then converges to the optimal infinite horizon policy π^*, as $t \to \infty$. Policy iteration algorithms usually consist of two stages: *policy evaluation* and *policy improvement*.

4.2.1 MDP policy iteration

First, we summarize the policy iteration method for MDPs [Howard, 1960]:

- Initialize $\pi_0(s) = a$, for all $s \in \mathcal{S}; a \in \mathcal{A}$ is an arbitrary action. Then, repeat the following policy iteration and improvement steps until the policy does not change anymore, i.e., $\pi_{t+1}(s) = \pi_t(s)$ for all states $s \in S$.
- Policy evaluation. Calculate the value of policy π_t (using Equation 18):

$$V^{\pi_t}(s) = R(s, \pi_t(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s, \pi_t(s), s')\, V^{\pi_t}(s').$$

- Policy improvement. For each $s \in \mathcal{S}$ and $a \in \mathcal{A}$, compute the Q-function $Q_t(s, a)$:

$$Q_{t+1}(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s')\, V^{\pi_t}(s'). \tag{55}$$

  Then, improve the policy π_{t+1}:

$$\pi_{t+1}(s) = \arg\max_{a \in \mathcal{A}} Q_{t+1}(s, a) \text{ for all } s \in \mathcal{S}. \tag{56}$$

Policy iteration tends to converge much faster than value iteration in practice. However, it performs more computation at each step; policy evaluation step requires a solution of a $|\mathcal{S}| \times |\mathcal{S}|$ linear system.

4.2.2 POMDP policy iteration

For value iteration, it is important to be able to extract a policy from a value function (see Section 3.3.1). For policy iteration, it is important to be able to represent a policy so that its value function can be calculated easily. Here, we will describe a POMDP policy iteration method that uses an FSC to represent the policy explicitly and independently of the value function.

The first POMDP policy iteration algorithm was described by Sondik [Sondik, 1971, 1978]. It used a cumbersome representation of a policy as a mapping from a finite number of polyhedral belief space regions to actions, and then converted it to an FSC in order to calculate the policy's value. Because the conversion between the two representations is extremely complicated and difficult to implement, Sondik's policy iteration is not used in practice.

Hansen proposed a similar approach, where a policy is directly represented by a finite state controller [Hansen, 1997, 1998a]. His policy iteration algorithm is analogous to the policy iteration in MDPs. The policy is initially represented by a deterministic finite-state controller π^0. The algorithm then performs the usual policy iteration steps: evaluation and improvement. The evaluation of the controller π is straightforward; during the improvement step, a dynamic programming update transforms the current controller into an improved one. The sequence of finite-state controllers $\pi_0, \pi_1, \ldots, \pi_t$ converges to the optimal policy π^* as $t \to \infty$.

4.2.3 Policy evaluation

In exact policy iteration, each controller node corresponds to an α-vector in a piecewise-linear and convex value function representation. Since our policy graph is deterministic, $\psi(n)$ outputs the action associated

with the node n, and $\eta(n, o)$ is the successor node of n after receiving observation o. The α-vector representation of a value function can be calculated using the cross-product MDP evaluation formula from before (Equation 48):

$$\bar{V}^{\pi}(\langle n, s\rangle) = R(s, \psi(n)) + \gamma \sum_{s', o} T(s, \psi(n), s')\, Z(s', \psi(n), o)\, \bar{V}^{\pi}(\eta(n, o), s'). \tag{57}$$

$\bar{V}^{\pi}(\langle n, s\rangle)$ is the value of state s of an α-vector corresponding to the node n:

$$\bar{V}^{\pi}(\langle n_i, s\rangle) \equiv \alpha_i(s). \tag{58}$$

Thus, evaluating the cross-product MDP for all states $\bar{s} \in \bar{\mathcal{S}}$ is equivalent to computing a set of α-vectors $\mathcal{V}^{\pi}$. Therefore, policy evaluation step is fairly straightforward and its running time is proportional to $|\mathcal{N} \times \mathcal{S}|^2$.

4.2.4 Policy improvement

Policy improvement step simply performs a standard dynamic programming backup during which the value function V^{π}, represented by a finite set of α-vectors $\mathcal{V}^{\pi}$, gets transformed into an improved value function V', represented by another finite set of α-vectors $\mathcal{V}'$. Although in the worst case the size of $\mathcal{V}'$ can be proportional to $|\mathcal{A}||\mathcal{V}^{\pi}|^{|\mathcal{O}|} = |\mathcal{A}||\mathcal{N}|^{|\mathcal{O}|}$ (where $|\mathcal{N}|$ is the number of controller nodes at the current iteration), many exact algorithms, such as Witness [Cassandra et al., 1994] or Incremental pruning [Cassandra et al., 1997], fare better in practice.

In the policy evaluation step, a set of α-vectors $\mathcal{V}^{\pi}$ is calculated from the finite-state controller π using Equation 57. Then, the set $\mathcal{V}'$ is computed using dynamic programming backup on the set $\mathcal{V}^{\pi}$. The key insight in Hansen's policy iteration algorithm is observation that the new improved controller π' can be constructed from the new set $\mathcal{V}'$ and the current controller π by following three simple rules:

- For each vector $\alpha' \in \mathcal{V}'$:
  - If the action and successor links of α' are identical to the action and conditional plan of some node that is already in π, then the same node will remain unchanged in π'.
  - If α' pointwise dominates some nodes in π, replace those nodes by a node corresponding to α', i.e., change the action and successor links to those of the vector α'.
  - Else, add a node to π' that has the action and observation strategy associated with α'.
- Prune any node in π that has no corresponding α-vector in $\mathcal{V}'$ as long as that node is not reachable from a node with an associated vector in $\mathcal{V}'$.

If the policy improvement step does not change the FSC, the controller must be optimal. Of course, this can happen only if the optimal infinite horizon value function does have a finite representation. Otherwise, a succession of FSCs will approximate the optimal value function arbitrarily closely; an ϵ-optimal FSC can be found in a finite number of iterations [Hansen, 1998b].

Like MDP policy iteration, POMDP policy iteration in practice requires fewer steps to converge. Since policy evaluation complexity is negligible compared to the worst-case exponential complexity of the dynamic-programming improvement step, policy iteration appears to have a clearer advantage over value iteration for POMDPs [Hansen, 1998a].

Controllers found by Hansen's policy iteration are optimized for all possible initial belief states. The convexity of the value function is preserved because the starting node maximizes the value for the initial belief state. From the next section onward, we will usually assume that an initial belief state is known

beforehand, and our solutions will take computational advantage of this fact. Optimal controllers can be much smaller if they do not need to be optimized for all possible belief states [Kaelbling et al., 1998, Hansen, 1998a].

5 Gradient-based optimization

Exact methods for solving POMDPs remain highly intractable, in part because optimal policies can be either very large, or, worse, infinite. For example, in exact policy iteration, the number of controller nodes might grow doubly exponentially in the horizon length; in value iteration, it is the number of α-vectors required to represent the value function that multiplies at the same doubly exponential rate.

An obvious approximation technique is therefore to restrict the set of policies; the goal is then to find the best policy within that restricted set. Since all policies can be represented as (possibly infinite) policy graphs, a widely used restriction is to limit the set of policies to those representable by *finite* policy graphs, or finite-state controllers, of some *bounded* size. This allows to achieve a compromise between the requirement that courses of action should depend on certain aspects of observable history, and the ability to control the complexity of the policy space.

Many previous approaches rely on the same general idea. While Hansen's exact policy iteration does not place any constraints on the policy graph structure, other techniques take computational advantage of searching in the space of structurally restricted FSCs. Littman [1994], Jaakkola et al. [1995], Baird and Moore [1999] search for optimal reactive, or memoryless, policies; McCallum [1995] considers variable-length finite horizon memory; Wiering and Schmidhuber [1997] attempt to find sequences of reactive policies; and, Peshkin et al. [1999] constrain the search to external memory policies. All of these techniques are special cases of searching in the space of finite policy graphs.

The restricted policy space that we will consider is representable by a limited size stochastic finite-state controller (see Section 3.4.1). Here, we describe the details of a gradient-based policy search method, introduced by Meuleau et al. [1999a,b]. The main idea of gradient-based POMDP policy search methods is to reformulate the task of finding optimal POMDP policies as a classical non-linear numerical optimization problem. If the stochastic FSC is appropriately parameterized so that its value is continuous and differentiable, the gradient of the value function can be computed analytically in polynomial time with respect to the size of the cross-product MDP ($|\mathcal{N} \times \mathcal{S}|$), and used to find locally optimal solutions.

5.0.5 Policy graph value

We can rewrite Equation 48, which calculates the value of a stochastic policy graph π, in a more concise matrix and vector form:

$$\bar{V}^{\pi} = \bar{R}^{\pi} + \gamma \bar{T}^{\pi}\, \bar{V}^{\pi}. \tag{59}$$

$\bar{V}$ and $\bar{R}$ are vectors of length $|\mathcal{N}|\,|\mathcal{S}|$, and $\bar{T}$ is an $|\mathcal{N}|\,|\mathcal{S}|$ by $|\mathcal{N}|\,|\mathcal{S}|$ matrix. Since $\bar{T}$ is a stochastic matrix and the discount factor $\gamma < 1$, the matrix $I - \gamma\bar{T}$ is invertible [Puterman, 1994]; we can thus solve Equation 59 for $\bar{V}$:

$$\bar{V}^{\pi} = (I - \gamma \bar{T}^{\pi})^{-1}\, \bar{R}^{\pi}. \tag{60}$$

Notice that $\bar{V}^{\pi}$, $\bar{T}^{\pi}$, and $\bar{R}^{\pi}$ depend on the policy graph $\pi = \langle \mathcal{N}, \psi, \eta \rangle$. Therefore, for a given number of nodes $|\mathcal{N}|$, the vector $\bar{V}^{\pi}$ could be optimized by choosing the right functions ψ and η. To convert this problem to a classical non-linear optimization problem, we need to make sure that the objective function is a scalar as well as appropriately parameterize the functions ψ and η.

5.0.6 Prior beliefs

The value vector $\bar{V}^{\pi}$ contains the total discounted cumulative reward for each system state s and graph node n. The total expected reward depends on the state and node in which an agent starts; this could be quantified by an agent's prior beliefs about the world. Let $\bar{b}_0$ be an $|\mathcal{N}|\,|\mathcal{S}|$ vector of probabilities representing the agent's prior beliefs about the states $\mathcal{S}$ and policy graph nodes $\mathcal{N}$. That is,

$$\begin{aligned} \sum_{n,s} \bar{b}(\langle n, s\rangle) &= 1, \\ \bar{b}(\langle n, s\rangle) &\geq 0 \text{ for all } n \in \mathcal{N}, s \in \mathcal{S}. \end{aligned} \tag{61}$$

Then, the total expected cumulative discounted reward E^{π} is just

$$E^{\pi} = \bar{b}_0 \cdot \bar{V}^{\pi}. \tag{62}$$

To simplify the problem, we will assume that the agent always starts in node n_0; it is a valid simplification if the initial policy graph structure is symmetric for all nodes. The agent's prior knowledge about the world is summarized by the belief vector b_0. Therefore,

$$\bar{b}(\langle n, s\rangle) = \begin{cases} b(s), & \text{if } n = n_0, \\ 0, & \text{otherwise.} \end{cases} \tag{63}$$

5.0.7 Soft-max parameterization

To parameterize the functions ψ and η, we will employ a commonly used *soft-max* distribution function [Meuleau et al., 1999b, Aberdeen and Baxter, 2002]. Let $\mathbf{x}^{\psi}$ and $\mathbf{x}^{\eta}$ be parameter vectors for the respective functions ψ and η. $\mathbf{x}^{\psi}$ will be indexed by a node n and an action a; $\mathbf{x}^{\eta}$ will be indexed by a node n, an observation o, and the successor node n'. We will use the notation $\mathbf{x}^{\psi}[n, a]$ to denote the ψ parameter indexed by n, a, and $\mathbf{x}^{\eta}[n, o, n']$ will be the η parameter indexed by n, o, n'. Then,

$$\psi(n, a) = \psi(a|n; \mathbf{x}^{\psi}) = \frac{e^{\mathbf{x}^{\psi}[n,a]}}{\sum_{\bar{a}\in\mathcal{A}} e^{\mathbf{x}^{\psi}[n,\bar{a}]}}, \tag{64}$$

$$\eta(n, o, n') = \eta(n'|n, o; \mathbf{x}^{\eta}) = \frac{e^{\mathbf{x}^{\eta}[n,o,n']}}{\sum_{\bar{n}'\in\mathcal{N}} e^{\mathbf{x}^{\eta}[n,o,\bar{n}']}}. \tag{65}$$

Because we use soft-max, the parameterized functions ψ and η still represent probability distributions; that is,

$$\begin{aligned} \sum_{a\in A} \psi(a|n; \mathbf{x}^{\psi}) &= 1, \\ \sum_{n'\in\mathcal{N}} \eta(n'|o, n; \mathbf{x}^{\eta}) &= 1, \\ \psi(a|n; \mathbf{x}^{\psi}) &\geq 0 \quad \text{for all } a \in \mathcal{A}, n \in \mathcal{N}, \\ \eta(n'|n, o; \mathbf{x}^{\eta}) &\geq 0 \quad \text{for all } n, n' \in \mathcal{N}, o \in \mathcal{O}. \end{aligned} \tag{66}$$

5.0.8 Objective function

Let $\mathbf{x}$ denote the combined vector of parameters $\mathbf{x}^\psi$ and $\mathbf{x}^\eta$. By substituting Equation 60 into 62, we finally get an unconstrained continuous objective function $f(\cdot)$ of parameters $\mathbf{x}$:

$$f(\mathbf{x}) = \bar{b}_0 \; (I - \gamma \bar{T}^\pi)^{-1} \; \bar{R}^\pi, \tag{67}$$

where (see Equations 49 and 50)

$$\bar{T}^\pi(\langle n, s\rangle, \langle n', s'\rangle) = \sum_{a,o} \psi(a|n; \mathbf{x}^\psi) \, \eta(n'|n, o; \mathbf{x}^\eta) \, T(s, a, s') \, Z(s', a, o), \tag{68}$$

$$\bar{R}^\pi(\langle n, s\rangle) = \sum_{a} \psi(a|n; \mathbf{x}^\psi) \, R(s, a), \tag{69}$$

and $\bar{b}_0, T(\cdot), R(\cdot), Z(\cdot)$ are supplied by the POMDP model. The number of parameters $|\mathbf{x}|$ depends on the POMDP model and the size of the policy graph (i.e., the size of the cross-product MDP):

$$|\mathbf{x}| = |\mathbf{x}^\psi| + |\mathbf{x}^\eta| = |\mathcal{N}||\mathcal{A}| + |\mathcal{N}||\mathcal{O}||\mathcal{N}|. \tag{70}$$

This presents two advantages to gradient-based methods of solving POMDPs: the number of parameters does not depend on the size of the state space $\mathcal{S}$, and the size of internal memory $\mathcal{N}$ can be controlled by a user.

5.0.9 Gradient calculation

Since the objective function $f(\mathbf{x})$ is a complicated series matrix expansion with respect to its parameters, function value based optimization techniques will be ineffective. To perform numerical optimization, we will need to employ first-order information about our objective function.

Because of the soft-max parameterization, the gradient of $f(\mathbf{x})$ can be calculated analytically. From Equation 62,

$$\frac{\partial f}{\partial x} \;\; = \;\; \bar{b}_0 \; \frac{\partial \bar{V}}{\partial x}. \tag{71}$$

From Equation 60,

$$\frac{\partial \bar{V}}{\partial x} = (I - \gamma \bar{T})^{-1} \; \left[\frac{\partial \bar{R}}{\partial x} + \gamma \frac{\partial \bar{T}}{\partial x} (I - \gamma \bar{T})^{-1} \; \bar{R} \right]. \tag{72}$$

Partial derivatives with respect to $\bar{T}$ and $\bar{R}$ can be calculated from Equations 68 and 69:

$$\frac{\partial \bar{T}}{\partial x^\psi} \;\; = \;\; \sum_{a,o} \frac{\partial \psi(a|n; \mathbf{x}^\psi)}{\partial x^\psi} \, \eta(n, o, n') \, T(s, a, s') \, Z(s', a, o), \tag{73}$$

$$\frac{\partial \bar{T}}{\partial x^\eta} \;\; = \;\; \sum_{a,o} \psi(n, a) \; \frac{\partial \eta(n'|n, o; \mathbf{x}^\eta)}{\partial x^\eta} \; T(s, a, s') \, Z(s', a, o), \tag{74}$$

$$\frac{\partial \bar{R}}{\partial x^\psi} \;\; = \;\; \sum_{a} \frac{\partial \psi(a|n; \mathbf{x}^\psi)}{\partial x^\psi} R(s, a), \tag{75}$$

$$\frac{\partial \bar{R}}{\partial x^\eta} \;\; = \;\; 0. \tag{76}$$

Finally, we can find the derivatives of ψ and η from the analytical expression of the soft-max function (see Equations 64 and 65):

$$\frac{\partial \psi(a|n;\mathbf{x}^{\psi})}{\partial x^{\psi}[\bar{n},\bar{a}]} = \begin{cases} (1-\psi(n,a))\,\psi(n,a), & \text{if } n=\bar{n},\ a=\bar{a}, \\ -\psi(n,a)\,\psi(\bar{n},a), & \text{if } n=\bar{n},\ a\neq\bar{a}, \\ 0, & \text{if } n\neq\bar{n}. \end{cases} \tag{77}$$

$$\frac{\partial \eta(n'|n,o;\mathbf{x}^{\eta})}{\partial x^{\eta}[\bar{n},\bar{o},\bar{n}']} = \begin{cases} (1-\eta(n,o,n'))\,\eta(n,o,n'), & \text{if } n=\bar{n},\ o=\bar{o},\ n'=\bar{n}', \\ -\eta(n,o,n')\,\eta(\bar{n},o,n'), & \text{if } n=\bar{n},\ o=\bar{o},\ n'\neq\bar{n}', \\ 0, & \text{if } n\neq\bar{n} \text{ or } o\neq\bar{o}. \end{cases} \tag{78}$$

The search for local minima can be performed using many numerical optimization techniques that employ the analytically calculated gradient information (such as steepest-descent, quasi-Newton or conjugate gradient).

References

Douglas Aberdeen and Jonathan Baxter. Scalable internal-state policy-gradient methods for POMDPs. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 3–10, 2002.

K. J. Aström. Optimal control of Markov decision processes with incomplete state estimation. *J. Math. Anal. Appl.*, 10:174–205, 1965.

Leemon Baird and Andrew Moore. Gradient descent for general reinforcement learning. *Advances in Neural Information Processing Systems 11*, 1999.

Richard E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, 1957.

Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1023–1028, Seattle, 1994.

Anthony R. Cassandra, Michael L. Littman, and Nevin L. Zhang. Incremental pruning: A simple, fast, exact method for POMDPs. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 54–61, Providence, RI, 1997.

Hsien-Te Cheng. *Algorithms for Partially Observable Markov Decision Processes.* PhD thesis, University of British Columbia, Vancouver, 1988.

Eric A. Hansen. An improved policy iteration algorithm for partially observable MDPs. In *Proceedings of Conference on Neural Information Processing Systems*, pages 1015–1021, Denver, CO, 1997.

Eric A. Hansen. Solving POMDPs by searching in policy space. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 211–219, Madison, WI, 1998a.

Eric J. Hansen. *Finite-memory control of partially observable systems*. PhD thesis, University of Massachusetts Amherst, Amherst, 1998b.

Ronald A. Howard. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, 1960.

Tommi Jaakkola, Satinder P. Singh, and Michael I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 345–352. The MIT Press, 1995.

Leslie Pack Kaelbling, Michael Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

Michael L. Littman. Memoryless policies: Theoretical limitations and practical results. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer, and Stewart W. Wilson, editors, *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, Cambridge, MA, 1994. The MIT Press.

Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable decision problems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 541–548, Orlando, 1999.

R. Andrew McCallum. Instance-based utile distinctions for reinforcement learning with hidden state. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 387–395, Lake Tahoe, Nevada, 1995.

Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R. Cassandra. Solving POMDPs by searching the space of finite policies. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 417–426, Stockholm, 1999a.

Nicolas Meuleau, Leonid Peshkin, Kee-Eung Kim, and Leslie Pack Kaelbling. Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 427–436, Stockholm, 1999b.

George E. Monahan. A survey of partially observable Markov decision processes: Theory, models and algorithms. *Management Science*, 28:1–16, 1982.

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

Leonid Peshkin, Nicolas Meuleau, and Leslie P. Kaelbling. Learning policies with external memory. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 307–314, San Francisco, CA, 1999.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.

Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

Edward J. Sondik. *The optimal control of partially observable Markov Decision Processes*. PhD thesis, Stanford university, Palo Alto, 1971.

Edward J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26:282–304, 1978.

C. C. White and D. Harrington. Application of Jensen's inequality for adaptive suboptimal design. *Journal of Optimization Theory and Applications*, 32(1):89–99, 1980.

Marco Wiering and Juergen Schmidhuber. HQ-learning. *Adaptive Behavior*, 6(2):219–246, 1997.