Hierarchical POMDP Decomposition for A Conversational Robot

Joelle Pineau and Sebastian Thrun
Carnegie Mellon University
Robotics Institute
5000 Forbes Avenue
Pittsburgh, PA 15213
jpineau@cs.cmu.edu, thrun@cs.cmu.edu

Abstract

POMDPs provide a useful framework for decision-making in the presence of uncertainty. Finding solutions to large-scale problems, however, has proven computationally infeasible. We propose a hierarchical approach to POMDPs which takes advantage of structure in the domain to decompose the problem into a collection of smaller POMDPs. These can be solved independently, allowing us to solve larger problems than were previously possible.

We apply this approach to the problem of human-robot speech dialogues, and show that appropriate decomposition can yield significant computational time reduction when finding a POMDP solution.

Introduction

Partially Observable Markov Decision Processes (POMDPs) provide a general decision-making framework for acting optimally in partially observable domains (Singh 1992; Kaelbling, Littman, & Cassandra 1998). POMDPs are well-suited to a great number of real-world problems where decision-making is required despite prevalent uncertainty, and as such have gained much attention in recent years (AAAI 1998). In a POMDP, a learner must select appropriate actions when operating in a stochastic environment where the state of the system is only partially observable. The goal is to find a policy that maximizes the reward obtained through the learner's choice of actions. Despite the representational power of POMDPs, their use is significantly limited by the great computational cost of finding an optimal policy for the learner. Consequently, recent efforts have focused on the development of efficient algorithms to generate near-optimal policies (Parr & Russell 1995). Current algorithms, however, are still generally unable to handle complex problems. We believe this can be overcome by exploiting the structure present in many real-world domains. This is in contrast to the conventional POMDP approach of treating the problem as a monolithic structure.

One such domain is that of dialogue models, which can possess a modular structure dictated by the arrangement of conversation topics. In this paper, we discuss how such structure can be exploited to solve large POMDPs more efficiently than is possible when using a conventional approach. Such structure is not present in all domains, but is found in those domains where there is some independence within the state space. In our approach, we decompose the POMDP problem into a collection of hierarchically-related smaller POMDP that can be solved independently using current solving algorithms. Similar work has been done for Markov Decision Processes (MDPs) (Dietterich 1998; Singh 1992; Dayan & Hinton 1993) and for Hidden Markov Models (HMMs) (Fine, Singer, & Tishby 1998). In the case of hierarchical POMDPs (HPOMDPs), the decomposition relies on the addition of "use_sub-module actions", which give higher-level POMDPs direct control over more specialized POMDPs. When a specialized POMDP is selected, it is identified as an "active module", and is responsible for the learner's action selection at a given time step. Other non-active modules act as HMMs during that time step, such that a belief state is still maintained over them. This ensures that these modules produce the appropriate behaviour when later selected as an "active module". The speedup obtained by applying our hierarchical approach to POMDPs can be remarkable in domains with appropriate structure.

The domain currently targeted by our work is that of dialogue management for speech-based interactions between a robot and a human user. This problem has recently received much attention (Asoh *et al.* 1999; Torrance 1994). Robot interactions with users are in many cases governed by manually scripted state-action sequences, which are generally not robust to unexpected events and require significant re-engineering given a change in the environment. Other work on modeling human-machine interactions as Markov processes has focused on using Markov Decision Processes (MDP), along with reinforcement learning, to generate optimal speech-based conversational interfaces (Singh *et al.* 1999; Levin, Pieraccini, & Eckert 1997). While this is a promising direction for the design of conversational interfaces, it is clear that the full observability assumption necessary for MDPs is not satisfied given the uncertainty inherent in today's speech recognition technology. POMDP models offer an appropriate framework

for robot-human dialogues, which we propose as an alternative to current dialogue management techniques. Our specific dialogue manager has been implemented for the verbal control of a mobile robot. The role of our conversational interface is mostly one of information-provider, which has a structure dictated by the information content available in a given application. We believe this domain can be naturally decomposed into a hierarchy and our approach can be used to solve large HPOMDPs in this context.

The next section presents an introduction to the HPOMDP structure, with a brief example illustrating the hierarchical decomposition of a simple model. The following section describes our current dialogue model, which is a complete example of using an HPOMDP as a conversational interface. Finally, we present a discussion of the important issues and future challenges presented by this work.

Hierarchical Decomposition of POMDPs

Review of POMDPs

POMDPs were developed to address the problem of choosing optimal actions in partially observable stochastic domains. The following definitions are necessary to an understanding of POMDPs:

- *States*: A set of states, $S = \{s_1, s_2, ..., s_n\}$, describes the problem domain. The domain is assumed to be in a specific state s_t at any point in time.
- *Actions*: A set of actions, $A = \{a_1, a_2, ..., a_m\}$, describes the agent's interaction with the domain. At any point in time the agent applies an action, a_t, through which it affects the domain.
- *Observations*: A set of observations, $O = \{o_1, o_2, ..., o_l\}$, is used to describe the agent's perception of the domain. A received observation, o_t, may only partially reflect the current state.
- *Rewards*: A set of numerical costs/rewards, $R(s_t, a_t, o_t, s_{t+1})$, are incurred during the agent's interaction with the domain. The rewards can be a function of the current state, selected action, received observation, and following state, however in most cases, rewards will not depend on all of these.

To fully characterize a specific POMDP model, the following probability distributions must be specified:

- $P(s) = Pr(s_0)$: An initial distribution of states at time $t = 0$. This is defined for all states in S.
- $T(s, a, s') = Pr(s_t = s' | a_{t-1} = a, s_{t-1} = s)$: A distribution describing the probability of transitioning from state s to state s', given that the agent selects action a. This is defined for all (s, a, s') triplets.
- $O(s, a, o) = Pr(o_t = o | s_t = s, a_t = a)$: A distribution describing the probability that the agent will perceive observation o, given that it is in state s, and has applied action a. This is defined for all (s, a, o) triplets.

At any given point in time, the system is assumed to be in some state s_t, which is never completely observable but is partially observable through observation o_t, therefore, in general, it is not possible to determine the current state with complete certainty. Instead, a *belief* distribution is maintained to succinctly represent the history of the agent's interaction (both applied and perceived) with the domain:

- $b_t = Pr(s_t | o_t, a_t, o_{t-1}, a_{t-1}, ..., o_0, a_0)$: The probability that, at time t, the agent is in state s_t, given the history $\{o_t, a_t, o_{t-1}, a_{t-1}, ..., o_0, a_0\}$. This distribution is defined over all states in S.

The goal of a POMDP is to learn an optimal policy describing action selection. An action is assumed to be optimal when, given the current belief vector b_t, it maximizes the expected discounted cumulative infinite reward. The *policy* can be defined as: $b_t \rightarrow a$, a mapping from belief state to action selection.

To operate in its domain and apply a policy, an agent must constantly update its belief vector. This can be done as follows:

$$b'(s') = Pr(s' | o, a, b) = \frac{O(s', a, o) \Sigma T(s, a, s') b(s)}{Pr(o | a, b)} \quad (1)$$

where the denominator is simply a normalizing factor. See (Kaelbling, Littman, & Cassandra 1998) for a more complete description of the POMDP problem formulation.

The Hierarchical POMDP Approach

The above section leaves open the question of finding an optimal policy. Given a fully specified POMDP model, there exists a number of techniques to solve it, and generate an optimal policy solution (Kaelbling, Littman, & Cassandra 1998). This solution specifies a mapping from all possible belief vectors to an optimal action. Finding such an optimal solution to a POMDP model, however, can be extremely computationally intensive, and is generally infeasible for anything but small models. This makes POMDP solutions impractical for many real systems.

We have addressed this problem by proposing a new approach that decomposes a POMDP problem into a hierarchy of related smaller POMDP models, each of which can be solved independently. Assuming that a domain can be modeled as a POMDP, we postulate that for certain domains, an equivalent HPOMDP can be obtained. Applicability of the HPOMDP approach is restricted to those domains which exhibit an appropriate structure. The necessary domain structure usually corresponds to the presence of localized regions in the state space, where some of the actions are only useful for a subset of the states. To state this more formally, given a POMDP model defined by $\{S, A, O\}$, a subset of states S' and actions A' can be extracted into a sub-model POMDP given the following condition:

- $\forall a_k \in A'$: $Pr(o | s_i, a_k) \neq Pr(o | s_j, a_k)$ OR $R(o, s_i, a_k) \neq R(o, s_j, a_k)$, for some $o \in O$ and for some $(s_i, s_j) \in S'$.

States in S' must therefore be distinguishable with respect to the actions in A', in terms of their observation probability and/or reward function. This is a necessary condition to the decomposition, which guarantees a valid POMDP is formed by the sets S', A', and O. It does not, however, guarantee that the decomposition will optimize the division of tasks between sub-models. Applying any of the decompositions satisfying this criteria will generate a valid HPOMDP solution. This can be similarly repeated to form other sub-models, given other sets S'' and A'' that satisfy the condition above.

Now assuming that such a new model has been defined, the hierarchical link between the original model $M=(S, A, O)$ and the new model $M'=(S', A', O)$, can be constructed by doing the following:

- remove the subset S' from S,
- remove the subset A' from A,
- add $s_k=child_k$ to S, where the $child_k$ state corresponds to the sub-domain M'
- add $a_k=use-child_k$ to A, where the $use-child_k$ action corresponds to selecting to use sub-model M'. This is referred to as a hierarchical action.

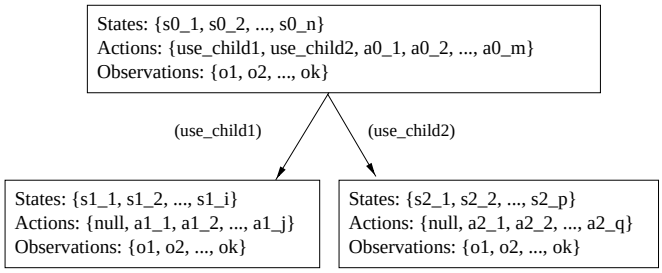

Figure 1: Sample Hierarchical Recursion

Figure 1 provides an illustration of a sample two-level hierarchy. The hierarchical decomposition is naturally recursive, therefore the simple example in figure 1 will serve to illustrate properties valid at all levels of a decomposition. Initially, we use a decomposition that takes the form of a tree-like hierarchy, without limitations with respect to depth. The decomposition postulates, at its simplest, that there is a parent-POMDP (e.g. top-node in figure 1), and any number of child-POMDPs (e.g. bottom-nodes in figure 1). The child-POMDP, assuming it is a bottom node and has no children of its own, behaves as a normal POMDP as defined by its own set of states S', actions A', observations O, and rewards. The parent-POMDP also behaves as a normal POMDP, with the added state and action described above. The parent-POMDP should have one such state s_k and action a_k for each child-POMDP. Its reward definition will specify the cost of taking action a_k when in state s_k as the expected value of the child-POMDP, as specified by its POMDP solution. The role of the parent-POMDP therefore includes, but is not limited to, the appropriate selection of a child-POMDP. Consequently, given a certain belief state, a policy for a parent-POMDP will dictate which child-POMDP it should invoke, or whether it should select a non-hierarchical action for example in an attempt to better determine which hierarchical action would be most appropriate.

Since all POMDPs in the decomposition are solved independently, they will each learn an independent optimal policy, which applies only to their respective subset of the state space. Before the application of these policies is discussed any further, the following four other key assumptions of the HPOMDP approach should be presented:

1. all POMDPs in the decomposition have an identical set of observations, O;
2. all POMDPs in the decomposition independently maintain their own belief vector;
3. all child-POMDPs have in their action set a *null-action*;
4. all belief vectors are independently updated given the latest *(action, observation)* pair.

To apply an overall policy, at any time t, the HPOMDP approach starts at the top-node of the hierarchy, applying that POMDP's optimal policy action given its current belief vector, this is recursively repeated down the tree as long as the policy action is a hierarchical action, specifiying the use of one of the child-POMDPs. This corresponds to following a path down the hierarchical structure, until a non-hierarchical action is selected. Following this, for each POMDP along that path, the respective belief vector is updated using the latest observation and their respective chosen action. For all POMDPs not along that path, the respective belief vector is updated using the latest observation and their respective *null-action*. The role of the *null-action* for each of the child-POMDPs is to allow them to act as HMMs, rather than POMDPS, whenever they are not selected as an active child-POMDP. Knowledge of all observations is maintained independently by all POMDPs, allowing them to react appropriately whenever called upon. Similarly, if there is switching between domains, previous knowledge is not lost, and need not be re-acquired when going back to a POMDP that was previously visited. This permits flexible switching between sub-domains, something that is often very difficult to realize with many hierarchical representations.

Decomposing a POMDP To verify that the HPOMDP approach yields an equivalent policy to that generated when using the full POMDP representation, a simple example is considered. This shows how, given an original single full-model POMDP representation, one can extract a subset of states to create a child-POMDP. We will compare the POMDP representation of the initial model, with the HPOMDP representation of the combined model, verifying that both representations yield the same policy. First we consider the fol-

lowing conventional POMDP representation of a simple three-state problem:
originalPOMDP: {
states: want-drink, want-apple, want-orange
actions: select-food-or-drink, select-fruit, get-drink, get-apple, get-orange
observations: none, drink, food, apple, orange
}

Alternatively, the model could be decomposed as:
parentPOMDP: {
states: want-drink, want-fruit
actions: select-food-or-drink, get-drink, use-child
observations: none, drink, food, apple, orange
}
childPOMDP: {
states: want-apple, want-orange
actions: null-action, select-fruit, get-apple, get-orange
observations: none, drink, food, apple, orange
}

Figure 2 illustrates the policy tree generated for the original model. Figure 3 shows the policy tree generated for the hierarchical model. One can clearly see that applying either policy will result in the same system behaviour, assuming the *use-child* action is transparent to the overall operation of the system. This is arguably not a useful application of hierarchy, but helps demonstrate that the combined policy obtained from the hierarchy is equivalent to the optimal policy of the original POMDP. For such a simple example, however, the benefits of the hierarchical representation are not apparent since using it requires more extensive model definition than does the conventional POMDP representation. The next section describes a model that we have implemented for which the hierarchical representation offers clear benefits.

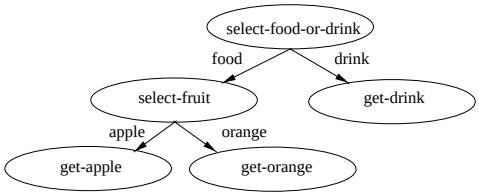

Figure 2: Sample POMDP Policy

Experimental Work

The current work stems from a project on the design of a conversation interface for a mobile robot. The goal is to develop a dialogue manager able to carry on a conversation with a user. This section briefly describes the robot interface and the HPOMDP model used in our dialogue manager, as well as presenting computation time results comparing POMDPs and HPOMDPs.

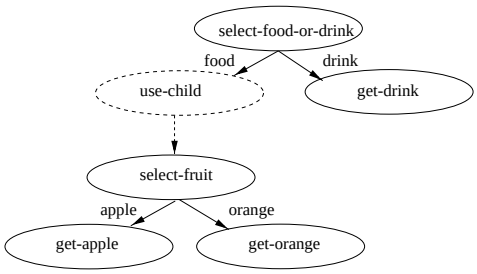

Figure 3: Sample HPOMDP Policy

The Robotic Application

Our robot, Flo, was designed as a preliminary prototype for a nursing assistant robot. The overall goal of the project is to develop personalized robotic technology that can play an active role in providing improved care and services to non-institutionalized elderly people. The target user would be an elderly individual with mild cognitive and/or physical impairement. Flo is a wheeled robot with an onboard speaker system and microphone for speech input and output. It also has two on-board PCs connected to the Internet via wireless Ethernet.

The current work focused on using the two-way speech interface as the basis for a POMDP-based dialogue manager. We opted for a POMDP representation for our dialogue model due to the representational power of POMDPs, and their ability to handle both observations and actions. This is essential to a dialogue manager such as ours, which must process a flow of inputs from the user and select appropriate responses based on these inputs. Initial work focused on designing the entire dialogue model as a single POMDP model, however the resulting model proved intractable for current POMDP solving techniques, unless reduced beyond the point at which it was useful as an interface manager. Using the hierarchical POMDP approach, however, it is possible to decompose the initial model into a number of smaller POMDP models designed to separately address the various components of the interface. The following section describes the HPOMDP representation used to implement our current dialogue manager.

The HPOMDP model

The current robot interface provides the user with practical information downloaded from the web. The present version offers information on three specific topics: the weather, the TV schedule and the calendar. For this application, the state set is designed such that it can represent the current state of the user in terms of his/her informational needs via the robot. The set of actions was selected such that it satisfies the user's informational requests. These take the form of various synthesized speech messages. Finally, observations correspond to the set of perceptual cues provided to the robot by the user. These consist of discrete spoken

words. These actions and observations are not an exhaustive list of the robot's communicative abilities, but cover the extent of actions and observations considered to date.

The hierachical decomposition was accomplished such that two of the information topics - the weather information, and the TV guide - are represented by two different POMDPs, each handling the conversation within the realm of their respective topic. A third POMDP is designed such that it governs the choice of sub-module at any point in time. The calendar topic is handled at the top level since it would be simply decomposed as a single-state, single-action domain. Figure 4 offers a summary illustration of the structure. This decomposition yields three different POMDPs which are small enough to be manageable by current solving and learning algorithms.

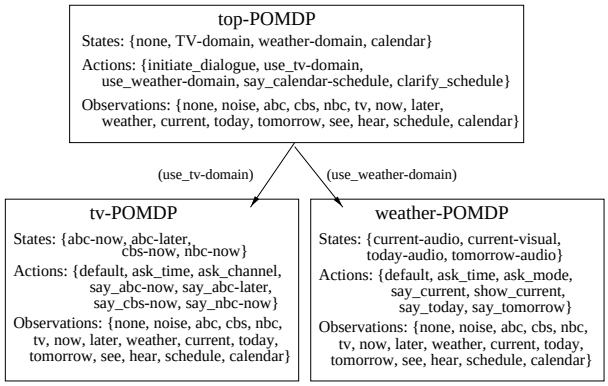

Figure 4: HPOMDP Dialogue Manager

Representing this domain as a single POMDP would require 10 states, 15 actions and 16 observations. Obtaining a solution for such a model was not computationally possible; we terminated the process when it failed to reach a solution after 24 hours of computation. Solving the HPOMDP model, however, required 480 seconds on a 400MHz Pentium II using the incremental pruning algorithm (Cassandra, Littman, & Zhang 1997). This was done using code provided by Tony Cassandra (Cassandra 1999).

The behaviour of the resulting system is illustrated by a conversational example, as detailed in Table 1. This shows how the tv-POMDP had already acquired knowledge of the tv station through the earlier observations, and therefore did not need to ask again for this information. Furthermore, the transition between the two submodels (tv guide and weather forecast) was accomplished smoothly. Even though all models are formulated in terms of the same complete set of observations, the models are specified such that observations irrelevant to a given model (E.g. "weather" when in tv-POMDP) are ignored by being assigned zero probability. It is interesting to note that despite the use of keywords as observations, ambiguous words (E.g. "schedule" - can refer to the TV schedule, or the appointment calender) are allowed, and can be disambiguated through proper action selection. This was done automatically when solving the POMDP models and obtaining a policy.

R: "How can I help you?"	$Act_{top-POMDP}$=Initialization
U: "What is playing on TV?"	$Observation$=tv
R: "On what channel?"	$Act_{top-POMDP}$=Use-tv $Act_{tv-POMDP}$=Select-channel
U: "On ABC."	$Observation$=abc
R: "For what time?"	$Act_{top-POMDP}$=Use-tv $Act_{tv-POMDP}$=Select-time
U: "Tell me the weather instead."	$Oberv ation$=weather
R: "For which day?"	$Act_{top-POMDP}$=Use-weather $Act_{weather-POMDP}$=Select-day
U: "How about today?"	$Observation$=today
R: "The forecast is rain."	$Act_{top-POMDP}$=Use-weather $Act_{weather-POMDP}$=Say-today
U: "Thanks, so about the TV..."	$Observation$=tv
R: "Yes, for what time?"	$Act_{top-POMDP}$=Use-tv $Act_{tv-POMDP}$=Select-time
U: "Now."	$Observation$=now
R: "ABC is now playing News at 6".	$Act_{top-POMDP}$=Use-tv $Act_{tv-POMDP}$=Say-abc-now
U: "Thanks."	$Observation$=(none)

Table 1: Sample Conversation with Flo

Results

To better understand the benefits of our approach, models of various sizes were generated, in both POMDP and HPOMDP form, to compare the computation cost of finding solutions to both. These results are presented in figure 5. The x-axis is an indication of increasing model complexity, as indicated by the number of states, actions and observations in the model. The numbers indicated reflect the size of the POMDP model; in the HPOMDP representation, each model is smaller, but the total number of states, actions and observations is greater due to the addition of some passive actions (E.g. $use - child_k$) and additional states (E.g. $subdomain_k$). The partial-HPOMDP line is generated based on models where only part of the domain was decomposed, therefore not taking advantage of the full decomposibility of that domain. In the illustrated cases, the HPOMDP consists of one parent-POMDP and two child-POMDPs, whereas the partial-HPOMDP consists of one parent-POMDP and only one child-POMDP. This graph shows that for very small domains, there is no benefits to the HPOMDP approach, however, the benefits increase dramatically for increasingly larger model size. For the HPOMDP cases, the

time shown is a sum of the times required to compute the solution for each module in the hierarchy. This could be further accelerated by some judicious use of parallel computing.

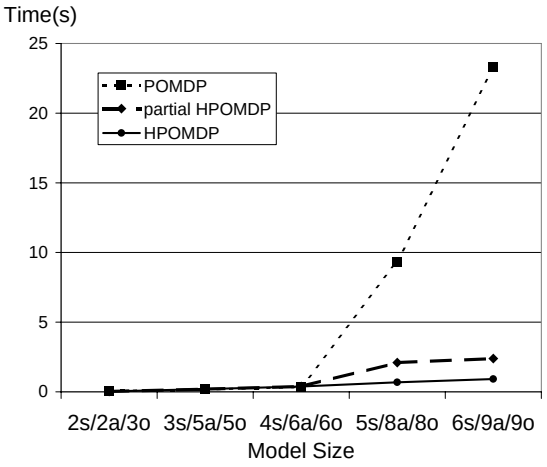

Figure 5: Comparing Computation Time

Discussion

We have presented a hierarchical approach to POMDPs which allows us to take advantage of structure present in the application domain, producing a representation that is easier to solve. We have shown how this can be applied to the dialogue management problem in the context of an interactive robot system. The results have demonstrated that the HPOMDP approach allowed us to obtain a policy much faster than with a non-hierarchical POMDP, and furthermore allowed us to build a larger POMDP-based dialogue manager than was possible with non-hierarchical POMDP representations.

The current HPOMDP implementation uses the expected value of the child-POMDP as the reward obtained by the parent-POMDP when invoking that child. This assumption will in some cases generate sub-optimal policies for the HPOMDP approach, most specifically in cases where the actual value obtained when using the child-POMDP is vastly different from the expected value. This has not occurred in any of the models we considered and/or implemented, however, could arise under certain conditions. We have not yet formalized the conditions under which sub-optimality arises. Future work will focus on formalizing the decomposition process, such that it can be performed automatically, as well as identifying necessary and sufficient conditions for which the policy generated by the HPOMDP is optimal.

References

1998. *AAAI Fall Symposium on POMDPs.* See http:/www.cs.duke.edu/ mlittman/talks/ pomdp-symposium.html.

Asoh, H.; Matsui, T.; Fry, J.; Asano, F.; and Hayamizu, S. 1999. A spoken dialog system for a mobile office robot. In *Proceedings of EuroSpeech'99*, 1139–1142.

Cassandra, A.; Littman, M. L.; and Zhang, N. L. 1997. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI'97)*, 54–61. San Francisco, CA: Morgan Kaufmann Publishers.

Cassandra, A. 1999. http://www.cs.brown.edu/ research/ai/pomdp/code/.

Dayan, P., and Hinton, G. 1993. Feudal reinforcement learning. In *NIPS 5*, 271–278. San Francisco, CA: Morgan Kaufmann.

Dietterich, T. G. 1998. The MAXQ method for hierarchical reinforcement learning. In *ICML*.

Fine, S.; Singer, Y.; and Tishby, N. 1998. The hierarchical hidden Markov model: Analysis and applications. *Machine Learning* 32.

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

Levin, E.; Pieraccini, R.; and Eckert, W. 1997. Learning dialogue strategies with the Markov decision process framework. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding.*

Parr, R., and Russell, S. 1995. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of IJCAI'95.* Montreal, Quebec: Morgan Kauffmann.

Singh, S.; Kearns, M.; Litman, D.; and Walker, M. 1999. Reinforcement learning for spoken dialogue systems. In *Proceedings of NIPS'99.*

Singh, S. P. 1992. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning* 8:323–339.

Torrance, M. C. 1994. Natural communication with robots. Master's thesis, MIT Department of Electrical and Computer Science, Cambridge, MA.