Hierarchical Methods for Planning under Uncertainty

Thesis Proposal

Joelle Pineau
Carnegie Mellon University
June 8, 2001

Abstract

The successful integration of robots in people's living environments strongly depends on robots' ability to effectively and meaningfully interact with humans. Human-robot communication requires that a robot be able to select appropriate actions in response to human speech or gestures. This role is played by the robot-based dialogue management system. Designing a dialogue management system is difficult primarily due to the rich multi-modal dimension of human communication, and the uncertainty inherent to the interaction. Uncertainty comes from many sources, including noise in the robot's sensors, changes in the environment, and the ambiguity of meaning in human communication. Furthermore, any model of the human-robot interaction upon which the dialogue management system is based is bound to be incomplete.

Probabilistic representations have evolved as the formalism of choice to accommodate stochasticity and uncertainty in AI systems. Partially observable Markov Decision Processes (POMDPs) specifically provide a useful framework for decision-making in the presence of uncertainty, and as such are well suited to address the dialogue management problem. However finding an optimal POMDP policy is computationally infeasible for large problems.

There is a strong belief within the AI planning community that by using structure inherent to a specific problem, one can appreciably reduce the computational complexity of solving that problem. I propose a hierarchical approach to POMDPs which takes advantage of structure in the domain to decompose the problem into a collection of smaller, hierarchically related, POMDPs. These can be solved independently, allowing us to find policies for complex tasks.

I present an algorithm to perform planning and policy execution for hierarchical POMDPs, and apply it to the robot dialogue management problem. Experiments with an implemented human-robot dialogue management system, as well as other simulation problems, show that the leveraging of hierarchical structure results in significantly reduced planning time, compared to a conventional POMDP, while the hierarchical policy still performs much better than a heuristic approximation.

1 Introduction

1.1 Motivation

The successful integration of robots in people's living environments strongly depends on a robot's ability to meaningfully interact with humans, while operating effectively in its environment. A robot-human dialogue management system is a specific type of interaction system. It is desirable for a robot deployed in a living environment because the interface feels natural to the human. However human behaviour exhibits unusual complexity, both because it encompasses a large spectrum of intentions, and because it can express these using rich multi-modal communication. Therefore the design of a robot dialogue system illustrates the difficulty of designing a system that can handle uncertainty. Designing robots that can operate successfully using only partial and noisy information remains a research challenge. In effect, we want a robot manager which can formulate an effective dialogue strategy, or plan, and apply it despite the prevalence of uncertainty in the task domain. Even when limiting the robot-human interaction to speech exchanges targeting a fixed set of conversation topics there is much uncertainty; speech recognition errors, ambiguous vocabulary, and poor auditory conditions all introduce uncertainty.

The type of uncertainty relevant to robot decision-making is three-fold. First, we are concerned with the uncertainty that arises when estimating the effects of a robot's action. Consider a robot that provides information on demand to a user, for example the weather forecast. By stating today's weather, the robot will potentially complete the task, but there is also the possibility that the user will want to inquire about further information. Second, we are concerned with the uncertainty that arises from partial or erroneous sensor measurements. Sensory information acquired through a speech recognizer is typically very noisy and can mislead the robot. Finally, any model of the human-robot interaction upon which the dialogue management system is based is bound to be incomplete. Therefore the third source of uncertainty we would like to accommodate is poor model parameterization. This thesis aims to propose new algorithmic planning approaches such that we can develop robot interface managers that are robust to these three types of uncertainty.

Probabilistic representations have evolved as the formalism of choice to accommodate stochasticity and uncertainty in AI systems [46]. Robot planning and execution in particular has been made more robust by the introduction of probabilistic representations such as Markov Decision Processes (MDPs), that accommodate stochastic state-to-state robot transitions, as well as probabilistic reward schedules. MDPs successfully address the first type of uncertainty.

To provide further robustness, Partially Observable Markov Decision Processes (POMDPs) have been introduced as an extension of MDPs, where the term *partially observable* refers to the fact that, in contrast with MDPs, we no longer assume that the state of the system is fully known at each time step. The decision-making is therefore conditioned on only partial information about the state of the world. POMDPs specifically extend MDPs to accommodate the second type of uncertainty, namely partial and noisy sensor information. However the increased representational flexibility of POMDPs is paralleled with a significant increase in computation complexity, compared to the simpler MDPs; most current POMDP planning algorithms can only solve problems with at most a few dozen states, which is clearly insufficient to model and control a robot-human dialogue manager, which may need a few hundred states to model meaningful interactions.

There is a strong belief within the AI planning community that by using structure inherent to a specific problem, one can appreciably reduce the computational complexity of solving that problem. More specifically, a "divide-and-conquer" approach can be adopted, where the inherent hierarchical structure of the problem domain is exploited to decompose the given problem into a set of related smaller problems (a.k.a. *subtasks*), all of which can be solved separately at much lower computational cost. The hierarchy typically takes the form of a recursive tree structure, where each node contains a distinct subtask. Previous work on Hierarchical HMMs, and Hierarchical MDPs [19, 18] have successfully shown that planning/learning using a hierarchical approach leads to reduced computation time, compared to a nonhierarchical counterpart. There are strong reasons to believe that the dialogue management problem offers the type of highly structured goals that can benefit from hierarchical partitioning.

1.2 Proposal

In this thesis, I propose the development of a new POMDP approach which allows us to do POMDP planning and execution for a large-scale robot interface manager, by exploiting hierarchical structure in the problem domain. This work directly addresses the current computational difficulties associated with using POMDPs,

I propose to apply hierarchical problem decomposition within the POMDP modeling framework, and present an algorithm that does planning by hierarchically decomposing a problem. The algorithm assumes that the problem is initially formulated as a conventional POMDP model, and its original action set is manually partitioned into a collection of hierarchically-related smaller subtasks. For each subtask, the algorithm independently learns a policy over its subset of actions using an existing POMDP solving algorithm.

Preliminary results show the effectiveness of the approach to control live human-robot dialogue models, as well as a variety of simulated POMDP problems containing other sources of sensory uncertainty. I also describe how abstraction can be applied within the hierarchical POMDP framework to reduce the state and observation sets, thereby substantially reducing overall planning time.

This proposal details how I intend to build on the existing work, to pursue two key research directions within the context of hierarchical POMDP planning. The first focus is on algorithmic design and analysis, as it pertainsto the hierarchical planning algorithm developed thus far. The current algorithm has been successfully applied to preliminary problems, however still lacks good mechanisms to propagate reward information between subtasks within the hierarchy. Additionally, I plan to offer theoretical bounds, which characterize the estimated value of applying the hierarchical POMDP policy.

The second focus is on learning, both of model parameters, and of the actual hierarchical structure. This addresses the third type of uncertainty listed earlier. For most robotic problems, it is unlikely that we would have access to a known, stationary POMDP model of the problem. More likely, we can postulate a partial or approximate model, which could be refined by further data collection. I plan to develop a learning algorithm that can interleave planning and parameter inference to incorporate new data into the model, and consequently produce a better policy. Similarly, the hierarchical partitioning of a problem into subtasks is currently done by hand, drawing on the designer's knowledge of the problem. I plan to look at heuristics that can be used to guide the hierarchical action partitioning, and apply these to automatically generate a hierarchical decomposition of a problem.

1.3 Solution Criteria

I plan to develop and implement a POMDP planning and execution approach that can be used to effectively control a robot's behaviour in the context of robot-human dialogue interactions.

This approach should exhibit the following characteristics:

- good performance demonstrated by the generation of near-optimal policies,
- large-scale applicability as indicated by the ability to handle realistic domains,
- adaptability to different users, acoustic conditions, and noise levels,
- flexibility to address other problem domains.

The successful completion of this research will lead to a new approach for planning using large-size POMDPs, which can be particularly effective for modelling highly complex, yet flexible, interaction systems. Other domains, such as robot navigation, will also be investigated to provide a deeper understanding of the generalizability of the hierarchical POMDP framework.

2 Related Work

2.1 Dialogue Management

Dialogue management is a critical component of a speech-based interface. The dialogue management problem refers to the decision-making required of an interactive system, to conduct successful two-way conversations between a user and the system. Assuming that the interactive conversation system can detect user utterances using automatic speech recognition, and can produce a response using text-to-speech technology, then the dialogue manager must decide what information to communicate to the user, as well as how and when to communicate it [59]. This decision can be complex, even for very simple interfaces, due to recognition errors, and untrained or uncooperative users. Of particular interest for this thesis is the problem of dialogue management for interactions between a robot and a human user. This problem has recently received much attention [2, 56].

Robot interactions with users are in many cases governed by manually-scripted state-action sequences [16, 57], which are generally not robust to unexpected events and require significant re-engineering given a change in the environment.

Recent 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 [30, 59, 20, 49, 58].

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. This strongly motivates the use of partially observable MDPs (POMDPs) for dialogue management.

2.2 Partially Observable Markov Decision Processes

POMDPs provide a general planning and decision-making framework for acting optimally in partially observable domains. They are well-suited to a great number of real-world problems where decision-making is required despite prevalent uncertainty. They generally assume a complete and correct world model, with stochastic state transitions, imperfect state tracking, and a reward structure, and from that find an optimal way to operate in the world.

They were first introduced by the operations research community, mostly through the work of Sondik, [51, 50, 52, 35], and typically focused on machine maintenance problems. The AI community has recently developed a growing interest in using POMDPs to solve a range of planning problems, including robot navigation [1, 47, 37, 45], robot interaction [13, 44], and medical diagnosis [40, 25], not to mention countless simulated maze-navigating agents.

I now review the numerous algorithms - both exact and approximate - that have been proposed to find POMDP policies.

2.2.1 Exact Value Function Algorithms

Sondik first proposed a value iteration algorithm, the One-Pass algorithm [51], which guaranteed convergence to an exact value function, and thereby an optimal policy, for finite horizon POMDPs. Many various algorithms building on the same idea have been proposed. Most of these exploit the fact that the POMDP value function is piecewise linear and convex, to reduce the computational cost of each value iteration. These include Monahan's Enumeration algorithm [35], the Witness algorithm [31, 27], and the Incremental Pruning algorithm [11]. For reasons detailed below (in section 3.2), these are completely impractical for problems with more than a few dozen states.

2.2.2 Policy search methods

The existing POMDP policy search methods are in many cases very similar to MDP policy search methods. The key idea is that for a given POMDP problem, one can define a space of valid policies, over which to apply a searching algorithm, alternating steps of policy evaluation, and steps of policy improvement.

Assuming a known model, with discrete state and actions sets, Hansen [22] proposed such a policy iteration algorithm for POMDPs. It typically performs faster than value iteration, but is still quite slow and unable to handle large problems. In large part, this is due to the dynamic programming update at each step, which is similar to that required in value iteration, though Hansen's policy iteration algorithm typically converges in less iterations than value iteration.

Most policy search algorithms focus on problems for which the model is known, and where the action space is continuous. A class of policies is defined, according to the action space parameterization, and the search is limited to that policy class. Kearns et al. [28] used a generative model of the POMDP to alternately build and evaluate trajectory trees. This approach was extended by Ng and Jordan [36], which reduced the number of trees required to guarantee a bound on the error of the policy's value.

Finally, making the strong assumption that the policy class is a differentiable function, Baxter and Bartlett [3] proposed a gradient ascent algorithm to optimize the policy.

These methods have shown significant promise, and have recently been used to solve interesting problems, albeit mostly in simulation. However it is unclear how well they will scale with problem size, specifically for problems with very large, non-parameterizable, action sets, as is characteristic of dialogue management domains.

2.2.3 Value approximation methods

Another class of proposed POMDP planning algorithm focuses on directly approximating the value function. Parr and Russell [38] solved discrete-state POMDPs by approximating the piecewise-linear continuous POMDP value function using a smooth and differentiable function that is optimized by gradient descent. Thrun [55] instead solved continuous-state POMDPs by using particle filtering to do approximate tracking of the belief state and using a nearest-neighbour function approximation for the value function. Neither of these has been applied to large-scale non-simulation problems.

2.2.4 Belief approach methods

There is yet another class of approximate POMDP algorithms which view a POMDP as a continuous-state MDP, and make various simplification assumptions in the belief state to reduce computational costs, trading off planning accuracy for speed. Hauskrecht [24] presents an excellent overview of various value-function methods, with a focus on belief-approximations. These methods use different heuristics to simplify the belief state and alleviate the computational costs of POMDP policy optimization.

Belief-approximation algorithms have been successfully applied to real-world problems. Nourbakhsh [37] used POMDPs for robot navigation, by approximating the POMDP policy using the optimal MDP policy, and assuming the "most-likely state heuristic". Koenig and Simmons [29] used POMDPs, including parameter training, for a robot navigation system. Roy et al. applied POMDPs to both robot navigation [45] and dialogue management [44]. This last approach uses a summary statistic of the belief state, consisting only of the most-likely state and the belief entropy, and optimize the MDP policy for this reduced representation of the belief state.

In all of these belief-approximation approaches, there is a clear tradeoff between sub-sampling the belief space, and the quality of the estimated value function. Furthermore these methods typically simplify the belief space using fixed schemes, rather than considering the problem domain and making domain-relevant belief simplifications.

2.2.5 Memory-based methods

Memory-based methods forgo a normal state representation, and instead use a policy that is conditioned on the most recent string of observations. McCallum [32] proposed an approach in which the observation histories are represented using a suffix tree with variable depth leaves, and where branches are grown whenever a new observation sequence is not Markovian with respect to the reward.

The memory-based approach does not rely on model parameterization, and is only subject to exploration. Consequently, a great amount of training data is required to learn a complete policy. This makes this approach problematic for planning problems where exploration costs are extremely high, as is the case with dialogue management.

2.3 Hierarchical MDP Methods

The existing planning POMDP methods are in general insufficient to handle realistic dialogue management problems in all their complexity. It is a strong assumption of this thesis that many real-world domains have structure that can be exploited to find good policies for complex problems.

The idea of leveraging structure to address large problems has been explored in Markov Decision Processes (MDPs) to solve complex problems. Most work in the area can be divided into two broad classes. In both cases, a single large MDP problem is treated as a collection of smaller MDP problems (i.e. subtasks) which can be solved independently. The first class, known as decomposition approaches, assumes that the set of subtasks can be treated as concurrent processes which are loosely coupled through shared resources, and a global policy is pieced together using the solution of each subtask. The second class, referred to as abstraction approaches, defines temporally abstract actions (which are in fact policies over primitive actions) and learns policies at different levels of action abstraction.

2.3.1 Decomposition Approaches

This class of approaches typically assumes that a given MDP problem can be decomposed into a set of related smaller MDP problems. The degree of coupling between these subtasks differs between approaches. In the simplest case, the state space is divided into subsets, each of which is assigned to a separate subtask, and these subtasks are coupled through interface states [15]. Therefore the coupling between subtasks is limited to transitions through the interface states and dependencies in the reward functions. In a second case, MDPs are decomposed along their action and/or state set, and subtasks are coupled through shared resources, where applying a given action in one subtask influences the resources available to other subtasks [6, 34, 48]. In a third case, MDPs are policy coupled, such that policy choices in one subtask can influences the states, actions, transition probabilities and rewards within other subtasks [60].

2.3.2 Abstraction Approaches

These algorithms leverage structure by imposing constraints on the set of acceptable policies. It is generally assumed that a global task can be completed by satisfying a number of subtasks. The policy constraints is a direct consequence of the subtask specification. Each subtask is typically represented by a reduced action set and specified termination conditions, though different algorithms have different requirements regarding the specification of subtasks. The "option" method [33] assumes each subtask is assigned a fixed policy by the designer. Parr and Russell's Hierarchy of Abstract Machines (HAM) [39] instead define each subtask using a non-deterministic finite-state controller. Finally, Dietterich's MAXQ method [17] defines each subtask using a termination predicate (e.g. end state) and a local reward function. Other work along this line includes [14]. Each of these three approaches uses a customized version of value iteration to optimize appropriate subtask selection.

The hierarchical POMDP algorithm I propose is closer, in terms of modelling and structural assumptions, to the abstraction approaches than to the decomposition approaches. However all these algorithms strongly rely on the assumption that the state of the environment, and consequently subtask completion, is fully observable. This is the most significant obstacle in extending these to partially observable domains.

2.4 Structured POMDP Approaches

There has been some preliminary work on exploiting structure in POMDP. One line of work has focused on using factored state representations to allow convenient representation of large problems. There has also been recent work specifically focused on exploiting hierarchical structure for POMDPs.

2.4.1 Factored State Representations

A common approach, used to succinctly represent a problem domain, is to specify a set of orthogonal multi-valued features to represent state, and/or action sets. Most POMDP policy-search algorithms target problems that can be represented using continuous parameterized state and action sets (see section 2.2.2). However Boutilier et al. [7, 5] have instead targeted problems that can be represented using a discrete set of actions, and a discrete state set. Their state space is fully defined by a set of state variables (e.g. robotInRoom, batteryStatus, etc.), each of which can take any of a number of discrete values (most often 0,1). They use two-state temporal Bayes nets with associated tree-structured conditional probability tables (CPTs) to describe the dynamics and rewards of the POMDP. The CPTs can be manipulated directly to perform exact value iteration or policy iteration [23]. Unfortunately, though this approach successfully reduces the POMDP state space representation, it does not directly reduce the size of the value function representation, which is a main obstacle to the efficient optimization of POMDP solutions.

2.4.2 POMDP hierarchical approaches

Preliminary attempts at hierarchical partitioning POMDP problems into subtasks typically make strict assumptions about prior knowledge of low-level tasks and ordering, which are substantially restrictive. The HQ-learning algorithm [61] learns a sequence of subgoals, assuming that each subgoal is satisfied through a reactive policy, and subgoal completion is fully observable. Castanon [12] addresses a very specific sensor management problem, in which he decomposes a multi-object detection problem into many single-object detection problems. He assumes a hierarchy of depth=2, where each single-object problem (i.e. low-level subtask) is solved using a standard POMDP algorithm, and these solutions are used to guide high-level coordination and resource allocation such that the multi-object problem is satisfied. This does not obviously extend to significantly different problems, such as those encountered in dialogue management.

Most recently, interesting hierarchical approaches to POMDP have been proposed which rely heavily on exploration and training to learn policies for large POMDP problems. Hernandez-Gardiol and Mahadevan [26] proposed a hierarchical extension of McCallum's Utile Suffix Memory algorithm [32], which builds observation histories at variable time resolutions. Theocharous and al. [54] extended the Hierarchical Hidden Markov Model [19] to include actions, and therefore accommodate POMDP problems, in fact allowing various levels of spatial abstraction. Both of these approaches assume that termination conditions are defined for subtasks, and can be detected during execution, which limits their applicability to many POMDP problems. Furthermore, they are best suited to problems where exploration and data collection is inexpensive, which is not the case with dialogue management.

3 Review of POMDPs

This section establishes the basic terminology used throughout this document, by providing a brief overview of the essential concepts in POMDPs (see [27] for a more complete description of the POMDP problem formulation).

A POMDP model, P, consists of:

- *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, $O = \{o_1, o_2, ..., o_k\}$, describes 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, $R(s_t, a_t, o_t, s_{t+1})$, describes the reinforcement received by the agent throughout its 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 (and typically only depend on the current state and the selected action).

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

- *Initial state probability distribution*:
$$P(s) := Pr(s_0) \tag{1}$$
the distribution describing the probability that the domain is in state s at time $t = 0$. This is defined for all states in S.

- *State transition probability distribution*:
$$T(s, a, s') := Pr(s_t = s' | s_{t-1} = s, a_{t-1} = a) \tag{2}$$
the distribution describing the probability of transitioning to state s', given that the agent is in state s and selects action a. This is defined for all (s, a, s') triplets.

- *Observation probability distribution*:
$$O(s, a, o) := Pr(o_t = o | s_t = s, a_t = a) \tag{3}$$
the 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 may not be completely observable, but is partially observable through observation o_t. 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' interaction (both applied and perceived) with the domain:

- *Belief*:
$$b_t := Pr(s_t | o_t, a_t, o_{t-1}, a_{t-1}, ..., o_0, a_0) \tag{4}$$
describes 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.

Given a POMDP model as defined above, there exist two interesting problems in POMDPs. The first is state tracking, that is, the computation of the belief state at each time step. The second is the optimization of an action selection policy.

3.1 State Tracking

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

$$b'(s') := Pr(s'|o, a, b) = \frac{O(s', a, o) \sum_{s \in S} T(s, a, s') b(s)}{Pr(o|a, b)} \tag{5}$$

where the denominator is simply a normalizing factor.

For most domains, this problem is trivial compared to that of computing a useful action selection policy. For the large scale problems I am interested in, the belief updating may become problematic, however this has been addressed by earlier work within the context of Bayesian dynamics networks [8, 41]. The work proposed in this thesis assumes that the real challenge is to find good policies for very large POMDP problems.

3.2 Computing a Policy

A POMDP policy, π, is defined as:

$$\pi : b_t \rightarrow a \tag{6}$$

a mapping from belief state to action selection. In this, POMDPs significantly differ from MDPs, where the policy is a mapping from state to action:

$$\pi : s_t \rightarrow a \tag{7}$$

The goal of planning, or POMDP problem solving, is to learn an action-selection policy that maximizes the (possibly discounted) sum of all expected future reward up to some time T:

$$E[\sum_{\tau=t}^{T} R(\tau)] \tag{8}$$

where $R(\tau)$ is the reward at time τ.

The most straight-forward approach to finding POMDP policies remains the value iteration approach, where iterations of dynamic programming are applied to compute increasingly more accurate values for each belief state b.

$$V : B \rightarrow R \tag{9}$$

After convergence, the value is the expected sum of all (possibly discounted) future pay-off the agent expects to receive up to time T, if the correct belief is b. A remarkable result in [51] shows that the value function is a piecewise linear, convex, and continuous function of the belief, with finitely many linear pieces.

POMDPs offer a powerful representational framework for planning under uncertainty. Unfortunately, however, the exact algorithms used to compute optimal policies are bounded by a doubly exponential computational growth in the planning horizon, and in practice can be exponential. Furthermore, a single step of value iteration is on the order of

$$|\Gamma_t| = O(|A||\Gamma_{t-1}|^{|O|}) \tag{10}$$

where $|\Gamma_{t-1}|$ represents the number of components necessary to represent the value function at iteration $t-1$. This points to the need for more efficient algorithms. Recent efforts have focused on the development of efficient algorithms to generate near-optimal policies (see section 2.2). However many approximate planning algorithms are driven by computational approximations, rather than by approximations that are guided by the problem itself. Namely, there has been very little work on exploiting the hierarchical structure natural to some problems, to facilitate their planning. This is the gap addressed by the proposed hierarchical POMDP planning and execution algorithm.

4 Hierarchical POMDPs

The "divide-and-conquer" approach has been used multiple times in AI to facilitate solving a single, large, complex problem by breaking it into numerous smaller, simpler, problems. This thesis proposes to follow this tradition by exploiting hierarchy in the POMDP framework.

The fundamental idea behind my approach is the decomposition of a POMDP problem based on an action hierarchy. Assuming a given fully-parameterized POMDP problem, the model designer hierarchically partitions its original action set such that it spans a collection of hierarchically-related smaller POMDPs, which I refer to as *subtasks* (I use notation $\mathbf{P}_i$ for a given POMDP subtask). Each action is assigned to one or more subtasks, where each subtask independently learns a policy over its subset of actions using existing POMDP solving algorithms. High-level subtasks generally learn policies over the selection of lower-level subtasks; and low-level subtasks are responsible for the selection of primitive actions.

4.1 Action Hierarchy

The defining element of this approach is the action hierarchy. Figure 1 illustrates the basic concept of an action hierarchy. Formally, an action hierarchy is a tree, where each leaf is labeled by an action $a \in A$, which is in essence a hierarchical partitioning of the original action set A for a given POMDP problem. Each action $a \in A$ (henceforth called *primitive actions*) must be attached to at least one leaf. The internal leaves are called *abstract actions* (I use a bar, as in $\bar{a}_i$, to indicate that an action is abstract.) Each $\bar{a}_i$ is in fact an abstraction of the actions in the nodes directly below it in the hierarchy (e.g. $\bar{a}_2$ is an abstraction of a_1, a_2, a_3.)

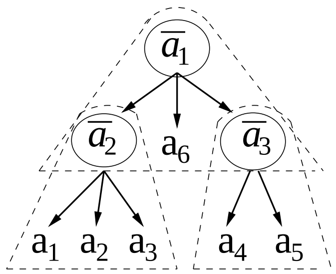

Figure 1: General Form Action Hierarchy

4.2 Task Decomposition

A key step towards hierarchical problem solving is to translate the action hierarchy into a collection of POMDPs that individually are smaller than the original POMDP, yet collectively define a complete policy.

In my approach, each internal node in the action hierarchy, together with its immediate children, defines a subtask $\mathbf{P}_i$ (shown as a triangle in Figure 1.)[1] Each subtask constitutes a separate POMDP, defined over the full state space S and observation space O, but where its set of applicable actions is limited to its immediate children in the action hierarchy. Policy optimization for each subtask is limited to its respective action subset. Figure 1 shows a problem that has been divided into three subtasks: $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$, with respective action sets: $\mathbf{P}_1 : A = \{\bar{a}_2, a_6, \bar{a}_3\}$, $\mathbf{P}_2 : A = \{a_1, a_2, a_3\}$ and $\mathbf{P}_3 : A = \{a_4, a_5\}$.

[1]The index for each subtask matches the index of the abstract action that spans that subtask.

4.3 Computing Local Policies

First, I assume that a conventional, fully-parameterized, POMDP model of the problem is specified by a designer. My approach then independently optimizes a local action policy for each subtask. A local action policy is a policy which is defined over a given action subset. It is crucial to note that only the subtasks with exclusively primitive actions (e.g. $\mathbf{P}_2$, $\mathbf{P}_3$) contain well-defined POMDP, where a well-defined POMDP is one for which all parameters (e.g. $T(s'|s,a), O(o|s,a), R(s,a)$) are defined. Subtasks containing abstract actions are so far ill-defined since the conventional POMDP does not provide meaningful parameters conditioned on these abstract actions. The hierarchical approach recursively makes use of the policies of lower-level subtasks to parameterize the abstract actions, and therefore proceeds in a bottom-up manner to find a local policy for each subtask, from the leaves of the hierarchy, to the root.

Subtasks with only primitive actions are solved first, using any of the existing POMDP planning algorithms (I currently use the incremental pruning algorithm, as described in [11], using code provided in [9]). Referring once again to figure 1, subtasks $\mathbf{P}_2$ and $\mathbf{P}_3$ can be solved right away, since the conventional POMDP model representation of the problem contains all necessary parameters.

To solve subtasks with abstract actions (e.g. $\mathbf{P}_1$), we need to infer model parameters $p(s'|s,\bar{a})$, $p(o|s,\bar{a})$ and $R(s,\bar{a})$ for all abstract actions. Let $\bar{a_i}$ be such an abstract action. Since the hierarchy is traversed in a bottom-up fashion, there already exists a POMDP solution for the subtask spanned by $\bar{a_i}$ (i.e. $\mathbf{P}_i$). Let π_{P_i} be the policy calculated for subtask $\mathbf{P}_i$. Then I define:

$$
\begin{aligned}
p(s'|s,\bar{a_i}) \;&:=\; p(s'|s,\pi_{P_i}(s)) && (11)\\
p(o|s,\bar{a_i}) \;&:=\; p(o|s,\pi_{P_i}(s)) && (12)\\
R(s,\bar{a_i}) \;&:=\; R(s,\pi_{P_i}(s)) && (13)
\end{aligned}
$$

where all the right-hand side terms are defined in the original POMDP model, and all the left-hand side terms are those needed to model the abstract action. In other words, $\bar{a_i}$ is parameterized on a state-by-state basis, using the action chosen by the policy of subtask $\mathbf{P}_i$. These definitions lead to a fully parameterized subtask, which can then be solved using any POMDP algorithm (I use [9]). Clearly, this definition of parameters constitutes an approximation. Consider for example subtask $\mathbf{P}_1$, which assumes parameters inferred for abstract actions $\bar{a_2}$, without having access to the full parameterization of actions $\{a_1, a_2, a_3\}$. Consequently, this approach cannot guarantee that a globally optimal policy will be found.

One important assumption of this approach is that each subtask must contain some amount of reward information, without which local policies cannot be meaningfully optimized. This is inconsistent with some single-goal problems where partial progress is not rewarded, however the variety of problems presented in the experimental sections suggests that many complex POMDP problems meet this requirement.

4.4 Calculating the Global Policy

The only remaining question is the construction of a global policy, using the set of local policies produced for the subtasks. In practice, the global policy is generated only at execution time. We first notice that all policies in the hierarchy are defined over the entire belief space, and assume that no abstraction is applied during belief tracking. Thus the agent is in possession of the full belief space.

To generate an action, the agent traverses the action hierarchy tree from the top to a leaf. At each level, the agent queries the local policy based on the current belief, and the action proposed by the policy is either primitive or abstract. If the action is a primitive action, it is directly executed by the agent. If the action is an abstract action, the agent queries the policy of the subtask spanned by this abstract action. It is trivial to show that this recursive algorithm always generates a primitive action.

The recursive action selection (and hierarchy traversal) is repeated at each time step. This differs from many hierarchical MDP algorithms where an agent 'remains' in a subtask until a so-called terminal state is reached. The difference is a consequence of the partial state observability in POMDPs, which suggests that we cannot guarantee detectability of terminal states.

4.5 State and Observation Abstractions

The hierarchical algorithm, as described so far, reduces the computational complexity of computing policies in POMDPs, specifically for large planning horizons. In particular, the number of linear pieces representing a conventional POMDP value function is recursively given by: $|\Gamma_t| = O(|A||\Gamma_{t-1}|^{|O|})$ (with $|\Gamma_0| = |A|$), which can be enumerated in time: $O(|S|^2|A||\Gamma_{t-1}|^{|O|})$. With the hierarchical POMDP approach, each POMDP subtask possesses a reduced action set, which factors in as an exponential factor in the overall complexity. These savings are partially offset by the fact that we now have to compute many policies, not just one.

Fortunately in many applications (including the ones discussed below), there is an opportunity to further reduce computational costs, specifically when considering domains where the difference between certain states is only relevant if a specific action is available. In this case, some state features may be safely ignored within certain subtasks (e.g. robot location can be ignored in some dialogue subtasks). Consequently the state set of each subtask can be reduced to include only relevant state features, as well as related observations. This is done on a subtask basis, where each can ignore those state features that are irrelevant to its small action subset.[2] Subtasks can therefore be defined over smaller state and observation spaces without influencing the policy optimization. The resulting computational savings can be tremendous (several orders of magnitude).

5 The Problem Domain

This section describes the problem domain that has motivated the algorithms proposed, and which will be used to demonstrate the usefulness of the approach.

5.1 Dialogue Management

A dialogue, D, can be formalized as a sequence of utterances:

$$D = \{M_0, U_0, M_1, U_1, ..., M_i, U_i, ..., M_N\} \tag{14}$$

where M_i and U_i are alternating machine utterances and user utterances [21]. Using this notion of a dialogue, a dialogue manager can be modelled as a policy function, π, which selects the machine utterance M_i based on the previous conversation history $\{M_0, U_0, ..., M_{i-1}, U_{i-1}\}$ and an internal knowledge-base (KB):

$$M_i = \pi(M_0, U_0, ..., M_{i-1}, U_{i-1}; KB) \tag{15}$$

This is directly aligned with the POMDP modelling framework, where the machine utterances M_i correspond to the POMDP action set, and the user utterances U_i correspond to the POMDP observation set.

The state is assumed to represent the user's conversational goal vis-a-vis the conversational system. As a result there is a very natural alignment between states and actions, which implies that a good MDP policy can often be easily generated by hand. However when designing a system with complex goals, mixed initiative, and unconstrained natural language, then even the MDP dialogue policy is typically not straightforward. The POMDP policy is further complicated by the need to plan for the entire belief state.

[2]This is currently done by hand, but should soon be automatic.

5.2 Robot Platform

The robot Pearl (shown in figure 2 interacting with some users) is the primary test-bed for the dialogue management system. It 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 is an elderly individual with mild cognitive and/or physical impairment. Pearl is a wheeled robot with an on-board speaker system and microphone, for speech input and output. It uses the Sphinx II speech recognition system [43] and the Festival speech synthesis system [4]. It also has two on-board PCs connected to the Internet via wireless Ethernet.

Figure 2: Pearl, the robotic nursing assistant, interacting with elderly users at a nursing facility

The robot can currently satisfy a varied set of tasks, including:

- *Intelligent reminding*: providing timely reminders to the user concerning a number of daily activities, (e.g. taking medication, ingesting food or water, etc.).
- *Data-collection and monitoring*: verifying the user's status (e.g. confirming whether medication has been properly taken).
- *Tele-presence*: recording and delivering audio-visual messages between the user and remotely-located medical personnel, or family members.
- *Information-provider*: providing the user with information upon request (e.g. weather forecast).
- *Social interaction*: carrying on some level of common social interaction (e.g. exchanging greetings).

In addressing the dialogue management problem, this thesis attributes an uncommonly large meaning to the word *dialogue*. The work initially focused on the development of a strictly speech-to-speech dialogue manager, however has evolved to include a variety of non-verbal cues. In essence, the dialogue manager has become a robot manager, and the robot behaviour governing the full set of activities described above is being modelled using the hierarchical POMDP approach. The exchanges between the robot and the user can exhibit significant uncertainty, in large part due to poor speech recognition. Therefore POMDPs are currently our best solution for this high-level robot control problem.

More specifically, the robot interface manager receives a wide range of inputs, including:

1. speech utterances from the user: E.g. *"Give me the weather forecast for tomorrow."*
2. commands from reminding module: E.g. *provide-medication-reminder-now*

3. sensor readings from the robot: E.g. *robot-is-in-kitchen*, *robot-battery-is-low*
4. manual operation commands: E.g. *remote-control-operation-requested*

These form the observation set for the POMDP model.

In response to these observations, the robot can apply one of many actions. Possible responses fall into four broad categories:

1. speech utterances sent to the speech synthesizer: E.g. *"Tomorrow's weather should be sunny, with a high of 80."*
2. motion commands sent to the navigation module: E.g. *send-robot-to-kitchen*
3. status information sent to the reminding module: E.g. *user-took-medication*
4. facemail operations: E.g. *record/play audio-visual messages*

These define the action set of the POMDP model.

Finally, the state set is spanned by the cross-product of many multi-valued state features, such as

1. robotLocation= {kitchen, living-room, bedroom},
2. robotDestination= {kitchen, living-room, bedroom},
3. userLocation= {kitchen, living-room, bedroom},
4. userGoal= {weatherToday, weatherTomorrow, sendRobotToKitchen, sendRobotToLivingroom, sendRobotToBedroom},
5. remindingModuleGoal= {medicationA, medicationB, activityA, activityB, activityC},
6. medicationStatus= {taken, not-taken, unknown},
7. batteryStatus= {high,low}.

The state features are selected to reflect the various possible goals that the robot can satisfy.

The model parameters are currently selected by a designer, though this thesis will eventually include a learning component for automatic model parameterization. The reward structure, also hand-crafted, reflects the relative costs of actions (e.g. robot motions actions are typically costlier than spoken verification questions), as well as the appropriateness of the action with respect to the state:

- positive rewards for correctly satisfying a goal
  E.g. R($a_i = send-robot-to-kitchen$)=+20
  if $s_i(robotDestination) = \{kitchen\}$, $s_i(robotLocation) = \{living-room, bedroom\}$;
- a large negative rewards for applying an action unnecessarily
  E.g. R($a_i = send-robot-to-kitchen$)=-100
  given state conditions other than above;
- a small negative reward for verification questions
  E.g. R($a_i = clarify-user-goal$)=-1
  given any state condition.

5.3 Other Domains

The robot interface problem is the main motivation behind this thesis. Though the hierarchical POMDP approach was developed specifically to address this problem, there are strong reasons to believe its usefulness far exceeds the robot interface domain. Other problems will be addressed to demonstrate this. These will mostly include well-known problems from the POMDP literature [10].

6 Preliminary Results

To evaluated my approach in practice, I applied the algorithm to three different simulation domains. I generated policies for all problems using three approaches: a conventional POMDP algorithm, my hierarchical POMDP algorithm (referred to as H-POMDP) and an MDP-solution, which solves the problem as an MDP and during execution uses the most likely state heuristic to map belief states to states.

Policy computations were performed using the incremental pruning algorithm for POMDPs, and value iteration for MDPs (all computations were performed on a 400MHz Pentium II). Tasks were evaluated using 5000 runs to show performance over time. The results reported were obtained using a simulated user due to the large number of experiments necessary to gain significance.

It is worth mentioning that the three simulation domains examined exhibit structure differently. In the first case, there is no structure to speak of. In the second case, a final goal can be satisfied through a sequence of intermediate subgoals. In the third case, the dialogue manager can satisfy alternate goals within a unified domain. Thus, the experiments address different problem setups always using the same algorithm, thereby shedding light onto our approach under different circumstances.

6.1 Simulation Domain 1 - Parts Manufacturing

The first task considered is the parts manufacturing problem introduced in Sondik's thesis [51]. I selected this problem specifically because it was not constructed to exhibit structure and can therefore illustrate the generality of the approach. Furthermore, I considered seven different hierarchical decompositions of this problem, to better study whether the algorithm is highly susceptible to a good design of the action hierarchy. Figure 3 shows one of the action hierarchy considered for this domain.

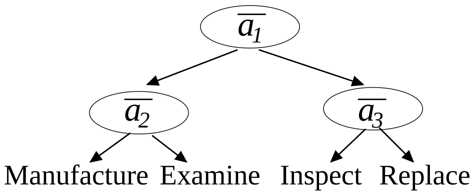

Figure 3: Action hierarchy for manufacturing task

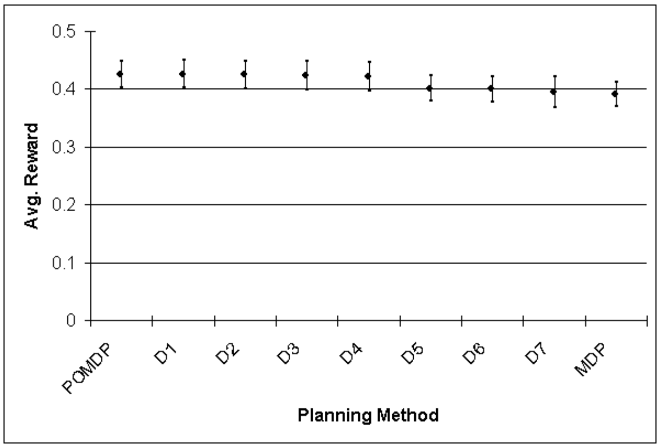

Figure 4: Manufacturing Task Results

Figure 4 illustrates the average reward for each of the seven decompositions (numbered D1-D7, in decreasing order of performance), as well as the average reward for the exact-POMDP and MDP-heuristic planners. In all H-POMDP cases (D1-D7), the average reward is larger than for the MDP case, though lower than for the POMDP case. It is worth mentioning that there is no theoretical guarantee that an H-POMDP policy will necessarily be better than an MDP policy.

It is interesting to note that there is a clear grouping amongst the seven H-POMDP decompositions. The first four decompositions yield near-optimal policies, whereas the last three are much weaker, though still marginally better than the MDP heuristic. I currently have no intuitive explanation for this performance difference, however it is the object of ongoing work.

6.2 Simulation Domain 2 - Navigation Task

The second task is a modified version of Dietterich's taxi task [17], with noisy perception. The problem is simple enough to be solvable using an exact POMDP solver. Figure 5 shows the action hierarchy for this domain.

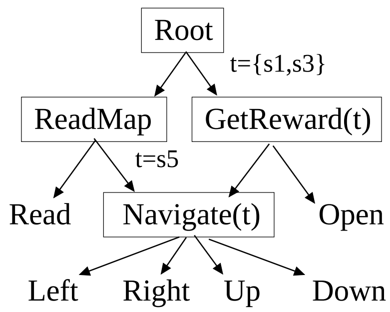

Figure 5: Action hierarchy for navigation task

Problem Solution $\lvert S\rvert$=11, $\lvert A\rvert$=6, $\lvert O\rvert$=6	CPU time (secs)	Average reward	% Goal success
MDP	0.000654	0.0	50
H-POMDP	2.84	12.2	100
POMDP	1119.93	12.5	100

Table 1: Performance Results for navigation domain

Table 1 presents results for this problem, showing policy computation times, and average reward per action. Performance with the hierarchical POMDP (H-POMDP) approaches that of the conventional POMDP (goal is always reached, but occasionally requires an extra action), and clearly exceeds that of the greedy MDP heuristic, which is unable to take advantage of uncertainty-reducing actions. Meanwhile, computation time with the H-POMDP is orders of magnitude faster than for the POMDP planner, yet orders of magnitude slower than with an MDP planner.

6.3 Simulation Domain 3 - Dialogue Management

The third task is a more challenging one. It is a simplified version of the robot interface domain described in section 5.2. Figure 6 shows the action hierarchy, which decomposes the action space along the natural divisions of various conversational goals.

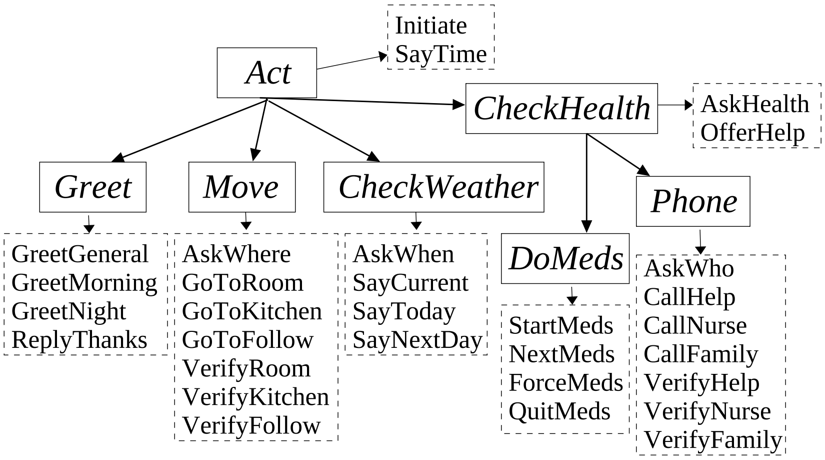

Figure 6: Action hierarchy for dialogue problem

Problem Solution $\lvert S\rvert$=20, $\lvert A\rvert$=30, $\lvert O\rvert$=27	CPU time (secs)	Average reward	% Goal success
MDP	6.46	53.33	80.0
H-POMDP	77.99	64.43	93.2
POMDP	24hrs+		

Table 2: Performance results for dialogue domain

Table 2 presents results for the dialogue problem, showing policy computation times, and average reward per action. Results show that even though the exact POMDP solver is unable to find a policy (and had only completed one iteration of the incremental pruning algorithm after 24 hrs), the hierarchical approach (H-POMDP) was able to obtain a policy in reasonable time. Execution performance with the H-POMDP policy was superior to that obtained with the MDP heuristic, both in terms of average reward per action and goal success rate. The performance difference between the MDP and H-POMDP policies is smaller for this domain than for the navigation task. This is mainly explained by the fact that in the dialogue domain, at many time steps the state is fully observable.

More extensive experiments are currently underway to verify the performance of the dialogue policy with real users. Table 3 provides a sample interaction in the dialogue domain, using the H-POMDP policy. It shows how the policy generated using this approach is able to make use of additional information-gathering actions, thereby allowing the system to recover from speech recognition errors (e.g. the third observation ("go_to") is erroneous: the speech recognition extracted "go to big" from the user's "go to room", and the H-POMDP decides that the cost of accidentally moving to the wrong location outweighs the cost of asking for clarification.) The MDP-heuristic policy did not exhibit similar adaptability to poor recognition performance.

Experimental work reported here constitutes positive evidence that the approach is useful for dialogue management. The next step is clearly to apply the approach to the full interface management domain described in section 5.2.

Actor	Speech	Observation	Action
Person	"Hello"	hello	
Robot	"Can I help you?"		Initiate
Person	"What time is it?"	time_request	
Robot	"It's now 11:41."		SayTime
Person	"Go to big."	go_to	
Robot	"Where should I go?"		AskWhere
Person	"Go to the."	go_to_room	
Robot	"Should I go to the room?"		VerifyRoom
Person	"Yes to the room."	go_to_room	
Robot	"You want me to go to the room?"		VerifyRoom
Person	"Yes."	yes	
Robot	*Robot goes to room*		GoToRoom

Table 3: An example dialogue from the interface domain.

7 Proposed Research Questions

This section details the research I propose to complete in support of this thesis. It is divided into four sections. The first two sections focus on the further development and analysis of the hierarchical POMDP algorithm described above. The last two sections focus on learning techniques, to be applied to the hierarchical POMDP model.

The section is organized as a list of questions designed to prompt future research. Each question is complemented by a short description of relevant background information motivating the question, as well as the specific task proposed to address this question, and a suggested initial approach to address the task.

7.1 Algorithmic Design and Implementation

7.1.1 Can we apply hierarchical planning to POMDPs?

Background: Many existing planning methods (both symbolic and probabilistic) have been extended to perform in a hierarchical setting. Typically, the planning algorithm is adapted such that a single planning problem can be treated as a hierarchy of smaller planning problems, thereby allowing planning for complex problems.
Task: Extend POMDP planning framework to allow planning and execution subject to a hierarchy of related POMDPs.
Approach: This task has been the subject of the initial work described in section 4 above. A hierarchical POMDP planning algorithm has been successfully implemented, and supplemented by a hierarchical POMDP execution algorithm. The algorithms have some limitations which are explored in the research questions listed below.

7.1.2 Can we use hierarchical POMDPs with a factorized model representation?

Background: A factorized state set is often the best way to describe a large planning problem. It describes the problem using a set of state and observation features which can take on any of a number of feature value. The full state (or observation) space is fully described by taking a cross-product of the state features and their respective values. This allows for a more succinct representation of the problem than would be possible when using a non-factorized model parameterization. It also assumes some independence between features (e.g. transition probabilities for a given action only affect a single state feature), and consequently can significantly simplify model parameterization for large problems.
Task: Implement a version of the hierarchical POMDP planning/execution algorithm that can handle factorized state and observation sets.

Approach: The initial implementation of the POMDP planning algorithm assumes a non-factorized state and observation sets. It could only practically be used for problems with at most 20-30 states and observations, mainly because of the need to specify by hand $O(|A||S|^2)$ state transition parameters and $O(|A||S||O|)$ observation probability parameters. A recent implementation (currently underway) uses factorized state and observation sets to allow more flexible model parameterization. This implementation was recently completed, and is currently used in ongoing experiments.

7.1.3 Can we accelerate hierarchical POMDP planning by applying abstraction?

Background: Many existing hierarchical planning approaches rely heavily on applying state/action abstraction to reduce the size of problems and thereby accelerate computation of good policies.
Task: Formalize the use of abstraction to reduce state, observation, and action sets. Derive conditions to apply correct abstraction (i.e. causing no loss of performance), and heuristics to apply approximate abstraction (i.e. causing minimal loss, possibly quantifiable).
Approach: Exact state abstraction and trivial observation abstraction have been included already in previous work (see section 4.5). Further observation abstraction may be possible. Approximate state abstraction has been addressed for MDPs in work by Boutilier et al., and could possibly be extended to POMDPs. Action partition is a fundamental assumption of the current action-driven hierarchical decomposition; it subsumes further action abstraction.

7.1.4 How can we best propagate value through the hierarchy?

Background: I currently use a heuristic approximation to set the value of low-level subtasks within the context of higher level subtasks.
Task: Find more accurate mechanisms to propagate value information between subtasks.
Approach: The first step is to look at how this is done in other work such as Dietterich's MAXQ, and Mahadevan's H-POMDP. None of these approaches are directly applicable to my hierarchical POMDP approach, however there may be interesting ways to extend their solution to my algorithm.

7.2 Algorithmic Analysis

7.2.1 Can we solve very large dialogue management problems using hierarchical POMDPs?

Background: The work done on the Pearl project represents the first known attempt to use POMDPs for dialogue management. The work has so far been limited to relatively small problems (at most 20-30 states, with similar numbers of actions and observations).
Task: Solve large dialogue management problems using the hierarchical POMDP approach, and use the resulting policy on a conversational robot.
Approach: The recent development of software to support a factorized model representation is essential to the successful completion of this task. I am in the process of applying this factorized representation version of the hierarchical POMDP approach to a problem with $4.9 * 10^6$ states (13 state features with between 2 and 13 values each), 47 observations, and 45 actions.

7.2.2 How does the use of hierarchy to facilitate POMDP planning compare to other approximate POMDP planning algorithms?

Background: There are many existing exact and approximate algorithms that can be used to solve POMDPs (e.g. witness, incremental pruning, "most-likely-state" MDP heuristic, augmented-MDP, policy search).
Task: Compare the performance of hierarchical POMDP planning and execution with that of other algorithms.

Approach: Select specific problems from [10], and solve them using the different algorithms. Compare the performance in terms of model complexity, planning time, and execution performance.

7.2.3 Can we quantify the loss introduced due to the hierarchical partitioning of the problem?

Background: By breaking down the problem into a set of related smaller problems, we ignore some dependencies between segments of the problem. This introduces a loss of performance.
Task: Propose bounds that characterize the performance loss introduced due to the hierarchy.
Approach: I currently introduce abstract actions into the problem formulation, to represent the effect of groups of low-level actions, at higher levels of the hierarchy. The parameters used to model these actions can be selected in different ways. When using a value-based approach for planning, I can select the abstract action parameters such that I can compute an upper-bound on the expected value of executing the hierarchical policy. This is subject to very restrictive additional conditions. I will look at ways of loosening these conditions, and see if I can similarly offer lower-bounds on the performance loss.

7.2.4 Can we perform a formal performance comparison of alternate hierarchical decompositions for a given POMDP problem?

Background: There may be many different ways to hierarchically decompose a given problem, either by hand or automatically.
Task: Find criteria to objectively compare alternate decompositions of a given POMDP problem.
Approach: Use metrics relating model complexity, planning time, performance loss, etc.

7.3 Hierarchical Structure Learning

7.3.1 Can we automatically hierarchically decompose a given problem?

Background: The hierarchical partitioning of a problem is typically done by hand, drawing on the designer's knowledge of the problem.
Task: Propose heuristics that can be used to automatically generate a hierarchical decomposition of a given POMDP problem.
Approach: Try grouping actions for which state transitions and observation probabilities affect the same subset of states. Look at the work done on automatically generating hierarchical decompositions for other planning problems (e.g. MDPs).

7.3.2 Given a learned policy for a hierarchical POMDP problem, can we change the hierarchical decomposition and re-learn a good hierarchical policy for the new decomposition, without fully re-planning?

Background: It may be difficult to commit to a single best hierarchical structure. Re-planning is necessary following changes to the hierarchical partition. However it may be possible to reduce computational load for re-planning by re-using some of the previous plan.
Task: Develop an algorithm that does re-planning following changes to the hierarchical action partitioning.
Approach: Assuming that the changes to the action partitioning are somewhat local (i.e. some parts of the action partition tree remain unchanged), re-planning is only necessary for those nodes of the tree which have a different set of children.

7.4 POMDP Parameter Learning

7.4.1 Can we learn POMDP model parameters?

Background: POMDP planning typically assumes that all model parameters (initial state, state transition and observation emission probabilities, as well as reward function) are known. The Baum-Welch algorithm has been extensively used to learn probabilistic parameters for HMMs [42], including hierarchical HMMs [19]. Similarly, many MDP learning algorithms have proposed to learn transition probabilities and the reward function [53].
Task: Find an algorithm to correctly learn new model parameters from training data.
Approach: There has been some work on extending the Baum-Welch algorithm to the POMDP framework [54], but it assumes that the reward function is known, and typically requires large amounts of data to produce good results. Other approaches should be pursued.

7.4.2 Given a learned policy for a POMDP problem, can we change the problem parameters and re-learn a good policy for the new parameters, without fully re-planning?

Background: This question applies to both the flat and hierarchical POMDP.
Task: Find algorithms to do re-planning following parameter changes.
Approach: For the flat POMDP representation, this problem can be tackled differently depending on whether planning is done using a policy-based approach, or value-based approach. Using policy methods, re-planning can be done by using the old optimal policy to initialize policy search. Using value methods, re-planning must be done from the beginning. For the hierarchical POMDP representation, the same is true, however there is the added benefit that re-planning may be unnecessary for some subtasks within the hierarchy, if all parameters of a given subtask have remained unchanged. In the case of local parameter learning, this may yield significant savings in re-plan time, especially if using a value-based planning approach.

8 Research Plan

8.1 Contributions

The scientific contributions of this thesis are two-fold. First, there are significant algorithmic developments, pertaining to hierarchical POMDP planning and execution. This will enable the use of POMDP for much larger problems than was previously possible. Second, the application of POMDPs to dialogue management is novel, and will allow the design of flexible and robust behavioural managers for interactive robots.

8.2 Schedule

This section describes the proposed timeline for the completion of this thesis.

Fall 01:

- Continue implementation of large-scale dialogue manager (tasks 7.1.1, 7.1.2, 7.2.1).
- Formalize conditions for state/observation abstraction, and implement relevant software (task 7.1.3).
- Investigate mechanisms to propagate value information through the hierarchy (task 7.1.4).

Spring-Summer 02:

- Generalize performance loss bounds (task 7.2.3).
- Develop framework to compare alternate hierarchical decompositions (task 7.2.4).
- Find heuristics to do automatic hierarchy decomposition (tasks 7.3.1, 7.3.2).

Fall 02:

- Perform formal comparisons with other POMDP approximation methods (task 7.2.2)
- Implement software to support POMDP parameter learning/adaptation (tasks 7.4.1, 7.4.2).
- Begin writing thesis document.

Spring-Summer 03:

- Finish experimentation with large-scale dialogue management.
- Complete thesis.
- Defend thesis.

References

[1] *AAAI Fall Symposium on Planning with Partially Observable Markov Decision Processes*, 1998. See http://www.cs.duke.edu /mlittman/talks/pomdp-symposium.html.

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

[3] J. Baxter and P. L. Bartlett. Reinforcement learning on POMDPs via direct gradient ascent. In *Machine Learning: Proceedings of the 2000 International Conference (ICML)*, 2000.

[4] A. Black, P. Talor, and R. Caley. The festival speech synthesis system. 1.4 edition, 1999.

[5] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[6] C. Boutilier, Brafman R. I., and Geib C. Prioritized goal decomposition of Markov decision processes: Toward a synthesis of classical and decision theoretic planning. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997.

[7] C. Boutilier and D. Poole. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of AAAI*, 1996.

[8] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 33–42, 1998.

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

[10] A. Cassandra, 1999. http://www.cs.brown.edu/ research/ai/pomdp/examples/index.html.

[11] A. Cassandra, M. L. Littman, and N. L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 54–61, 1997.

[12] D. A. Castanon. Approximate dynamic programming for sensor management. In *Conference on Decision and Control*, 1997.

[13] T. Darrell and A. Pentland. Active gesture recognition using partially observable markov decision processes. In *Proceedings of 13th IEEE Intl. Conference on Pattern Recognition (ICPR)*, 1996.

[14] P. Dayan and G. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 5, pages 271–278, San Francisco, CA, 1993. Morgan Kaufmann.

[15] T. Dean and S. H. Lin. Decomposition techniques for planning in stochastic domains. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*, 1995.

[16] M. Denecke and Waibel A. Dialogue strategies guiding users to their communicative goals. In *Proceedings of Eurospeech*, 1997.

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

[18] T. G. Dietterich. State abstraction in MAXQ hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 12, pages 994–1000, 2000.

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

[20] J. C. Fromer. Learning optimal discourse strategies in a spoken dialogue system. Master's thesis, MIT, 1998.

[21] D. Goddeau and J. Pineau. Fast reinforcement learning of dialog strategies. In *Proceedings of ICASSP*, 2000.

[22] E. A. Hansen. Solving POMDPs by searching in policy space. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 211–219, 1998.

[23] E. A. Hansen and Z. Feng. Dynamic programming for POMDPs using a factored state representation. In *Fifth International Conference on Artificial Intelligence Planning and Scheduling*, 2000.

[24] M. Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[25] M. Hauskrecht and H. Fraser. Planning medical therapy using partially observable markov decision processes. In *AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes*, 1998.

[26] N. Hernandez-Gardiol and S. Mahadevan. Hierarchical memory-based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 13, 2001.

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

[28] M. Kearns, Y. Mansour, and A. Y. Ng. Approximate planning in large POMDPs via reusable trajectories. *Advances in Neural Information Processing Systems*, 12, 1999.

[29] S. Koenig and R. G. Simmons. Unsupervised learning of probabilistic models for robot navigation. In *Proceedings of the International Conference on Robotics and Automation*, 1996.

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

[31] M. L. Littman. *Algorithms for Sequential Decision Making*. PhD thesis, Brown University, 1996.

[32] A. K. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1996.

[33] A. McGovern, D. Precup, B. Ravindran, S. Singh, and R. S. Sutton. Hierarchical optimal control of MDPs. In *Proceedings of the 10th Yale Workshop on Adaptive and Learning Systems*, pages 186–191, 1998.

[34] N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L. P. Kaelbling, T. Dean, and C. Boutilier. Solving very large weakly coupled markov decision processes. In *Proceedings of AAAI'98*, 1998.

[35] G. E. Monahan. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, January 1982.

[36] A. Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.

[37] I. Nourbakhsh, R. Powers, and S. Birchfield. Dervish: An office-navigation robot. *AI Magainze*, pages 53–60, Summer 1995.

[38] R. Parr and S. Russell. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*, Montreal, Quebec, 1995. Morgan Kauffmann.

[39] R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems*, volume 10, 1998.

[40] N. Peek. Predictive probabilistic models for treatment planning in paediatric cardiology. In *AAAI 1998 Fall symposium on Planning with Partially Observable Markov Decision Processes*, 1998.

[41] P. Poupart and C. Boutilier. Value-directed belief state approximation for POMDPs. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.

[42] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–285, 1989.

[43] M. Ravishankar. *Efficient Algorithms for Speech Recognition*. PhD thesis, Carnegie Mellon, 1996.

[44] N. Roy, J. Pineau, and S. Thrun. Spoken dialog management for robots. In *Proceedings of the 38th Annual Meeting of the Association for Computation Linguistics*, 2000.

[45] N. Roy and S. Thrun. Coastal navigation with mobile robots. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

[46] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[47] R. Simmons and S. Koenig. Probabilistic navigation in partially observable environments. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

[48] S. Singh and D. Cohn. How to dynamically merge markov decision processes. In *Advances in Neural Information Processing Systems*, volume 10, pages 1057–1063, 1998.

[49] S. Singh, M. Kearns, D. Litman, and M. Walker. Reinforcement learning for spoken dialogue systems. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

[50] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[51] E. J. Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, 1971.

[52] E. J. Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations Research*, 23(2):282–304, March-April 1978.

[53] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[54] G. Theocharous, K. Rohanimanesh, and S. Mahadevan. Learning hierarchical partially observable Markov decision process models for robot navigation. In *Proceedings of ICRA*, 2001.

[55] S. Thrun. Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

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

[57] M. Walker, D. Hindle, J. Fromer, DiFabbrizio G., and Mestel C. Evaluating competing agent strategies for a voice email agent. In *Proceedings of Eurospeech*, 1997.

[58] M. A. Walker. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, 12:387–416, 2000.

[59] M. A. Walker, Fromer J. C., and Narayanan S. Learning optimal dialogue strategies: A case study of a spoken dialgue agent for email. In *Proceedings of the 36th Annual Meeting of the Association for Computation Linguistics*, 1998.

[60] G. Wang and S. Mahadevan. Hierarchical optimization of policy-coupled semi-markov decision processes. In *Machine Learning: Proceedings of the 1999 International Conference (ICML)*, 1999.

[61] M. Wiering and J. Schmidhuber. HQ-learning. *Adaptive Behavior*, 6(2), 1997.