A Heuristic Search Algorithm for Acting Optimally in Markov Decision Processes with Deterministic Hidden State

Jamieson Schulte and Sebastian Thrun
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{*jschulte,thrun*}@*cs.cmu.edu*

Abstract

We propose a heuristic search algorithm for finding optimal policies in a new class of sequential decision making problems. This class extends Markov decision processes by a limited type of hidden state, paying tribute to the fact that many robotic problems indeed possess hidden state. The proposed search algorithm exploits the problem formulation to devise a fast bound-searching algorithm, which in turn cuts down the complexity of finding optimal solutions to the decision making problem by orders of magnitude. Extensive comparisons with state-of-the-art MDP and POMDP algorithms illustrate the effectiveness of our approach.

1 Introduction

The topic of decision making under uncertainty has received considerable attention in the past few years. Based on the recognition that uncertainty arises in a huge number of real-world problems, recent research has led to a range of paradigms for decision making under uncertainty. Today's two most prominent such paradigms are known as Markov decision processes (MDPs) [16] and partially observable Markov decision processes (POMDPs) [5]. While MDPs can cope with uncertainty regarding the effects of an agent's actions, POMDPs also address uncertainty in perception and uncertainty arising from hidden state.

In many robotics domains, neither of these paradigm suits well. MDPs are too specific in their assumption of full observability. POMDPs, on the other hand, are so general that today's best algorithmic solutions can only cope with a dozen states or so. This raises the question as to whether there exist intermediate problem formulations, which address uncertainty yet allow for algorithms that scale better to larger state spaces.

This paper explores a family of problems that extend MDPs with a limited type of hidden state (without noise in perception). We postulate the model of *Markov decision processes with deterministic hidden state (MDPDHS)*, which augments MDPs with hidden state variables that influence all observables in *deterministic* ways. The model captures important practical problems currently not contained in the MDP model, such as the mobile robot exploration problem [13], certain dialogue management problems [14], various mobile robot navigation problems in which environments possess detectable state, such as doors that might be open or closed [3], and graph-theoretical problems such as the Canadian Travelers Problem, which is the problem of traveling through a graph where edges might be unpassable [10].

We propose a heuristic search algorithm for finding optimal policies in MDPDHSs. Our approach performs heuristic search in the space of all information states, similar to approaches popular in the POMDP literature [5], but in a discrete way. The primary algorithmic contribution of this paper is a mechanism for accelerating the search that owes its validity to the specific nature of the hidden state in MDPDHSs. This mechanism uses fast state-based search to bound the goodness of information states. The bounds are used to prune the search tree, and they are used as a search heuristic for determining which information state to expand next in the search. Extensive comparisons with both the MDP and the POMDP solutions illustrate the superiority of our algorithm for this specific problem class.

By proposing the MDPDHS model and showing that model-specific insights can be used to design efficient algorithms, we hope to stimulate research on decision making that incorporates hidden state but not at the full generality of the POMDP model—which appears to be intractable in many real world problems.

2 Markov Decision Processes with Deterministic Hidden State

The MDPDHS model (short for: *MDPs with Deterministic Hidden State*) lies between classical Markov decision process (MDPs) and partially observable Markov decision process (POMDPs). This section describes the basic MDPDHS framework, beginning with a brief review on MDPs.

2.1 "Classical" Markov Decision Processes

A Markov Decision Process (MDP) consists of the following components:

- **States.** The state of the MDP is denoted by x. At any point in time, the state is fully observable. Put differently, there is no notion of partial observability, hidden state, or sensor noise in MDPs.
- **Actions and state transitions.** Actions, denoted by u, affect the state of the MDP in a probabilistic way. State transitions are characterized by a time-invariant conditional probability distribution $p(x'|u,x)$ that characterizes the probability of being in state x' after executing action u in state x.
- **Reward.** MDPs require a reward function $R(x) \longrightarrow \Re$, which measures the 'goodness' of each state x.

The central problem in MDPs is to devise a policy for action selection $\pi(x) \longrightarrow u$ that maximizes the expected cumulative future reward:

$$E[\sum_{t=0}^{\infty} \gamma^t R(x(t))] \tag{1}$$

Here $0 < \gamma \leq 1$ is a discount factor, which may be used to decay the reward over time. E denotes the mathematical expectation, and t is a time index. A well-known difficulty of finding such a policy arises from the fact that actions might affect rewards many time steps later. Today's most popular methods for calculating such policies are based on *value iteration* [16], *policy iteration* [17], or perform the search directly in policy space using gradient descent [2, 8]. The relation to heuristic search was pointed out in [6]. However, the main deficiency of the MDP model in many real-world domains (such as robotics) is the requirement for fully observable state.

2.2 Deterministic Hidden State

The MDPDHS model allows for the existence of a restricted type of hidden state. The nature of hidden state and its effect on state transitions and perception is limited in a way that allows for efficient, discrete problem solving, making the model similar to MDPs. In particular, it is not as general as the POMDP family of models [15, 5]. Nevertheless, as we shall see below, it is powerful enough to capture a range of practical problems with hidden state.

MDPDHSs extend MDPs as follows:

- **Hidden states**. In addition to the observable state x in MDPs, MDPDHSs possess hidden state, denoted by y. The state space of an MDPDHSs, thus, is the cross product of the observable state x and the hidden state y. Initially, the hidden state is unknown to the agent. Instead, it is given a prior probability distribution $p(y)$ over the values of the hidden state.
- **Hidden state transitions.** Hidden state transitions are deterministic—hence the term 'deterministic' in MDPDHSs. They are governed by a state transition function $\sigma(x,y,u) \longrightarrow y$. In addition, if the hidden state influences the transition of a fully observable state variable in x, this transition must also be deterministic (and such transitions must exist, otherwise the hidden state is irrelevant).
- **Measurements.** To infer information regarding the hidden state y, the agent can sense. Sensor measurements will be denoted by z. Measurements of state variables in x and y are also also deterministic, that is, they are governed by a function $\mu(x,y) \longrightarrow z$.

It is easy to see that the MDPDHS model is a generalization of the MDP model. The key advantage of the MDPDHS model is that it allows for hidden state—which is paramount for many real world applications. However, from a technical point of view any MDPDHS can be transformed into an MDP of possible exponential size (see below). By doing so, one loses important insights into the structure of the problem on which the algorithm described in this paper builds. We also note that the MDPDHS model is not as general as the POMDP model. For example, POMDPs allow for noise in perception, whereas MDPDHSs do not.

2.3 Information States in MDPDHSs

The key advantage of the MDPDHS model—which our algorithm below exploits—is the fact that the *information state space* is *finite*. The finiteness of the information state space implies that discrete search algorithms can be applied to find optimal policies in MDPDHSs.

Following [5], the *information state* is the state of knowledge an agent might have regarding the true state of the world, x and y. In the case of the observable state x, the corresponding part of the information state is simply equivalent to x. For the hidden state y, however, the corresponding part of the information state is a probability distribution over all possible states y [15]. This distribution will be denoted p_I, to indicate that it corresponds to an information state. The complete information state is thus the tuple $\langle x; p_I \rangle$.

As the agent acts and perceives, its information state changes. The observable state x changes stochastically according to the state space dynamics $p(x'|u, x)$. The change of the distribution p_I is captured by the popular Bayes filters [4], applied to the restricted state space dynamics in MDPDHSs. In particular, state transitions modify the information state as follows:

$$p_I(y = a) \quad \longleftarrow \quad \sum_b I[\sigma(x, b, u) = a]\; p_I(y = b) \tag{2}$$

Here x is the present observable state, u is the action, $I[\ ]$ denotes the indicator function, and the variables a and b instantiate possible hidden states y. Measurements z modify the information state as follows:

$$p_I(y = a) \quad \longleftarrow \quad \eta\; I[\mu(x, a) = z]\; p_I(y = a) \tag{3}$$

Here η is a normalizer that comes straight out of Bayes rule.

At first glance, it might not be obvious why the information state space in MDPDHSs is finite. In principle, any distribution over $p_I(x)$ constitutes a valid information state with respect to the hidden variables y—and the space of all distributions if infinite. However, MDPDHSs owe the finiteness of their information state space to their deterministic dynamics and measurement functions relative to the hidden state y.

So why is the information state space in MDPDHSs finite? Obviously, each information state p_I is defined through the values $\{p_I(y_1), p_I(y_2), \ldots, p_I(y_m)\}$, with m being the size of the hidden state space. Deterministic state transitions can only permute these values or add some of them together. For example, the two values $p_I(y)$ and $p_I(y')$ with $y \neq y'$ are added together if $\sigma(x, y, u) = \sigma(x, y', u)$. Consequently, deterministic state transitions can only lead to a finite number of successor distributions, since there are only finitely many permutations and there may only be a finite number of cases where two or more non-zero p_I-values are added together. A similar statement can be made for measurements. Each value $p_I(y)$ will either remain the same (before normalization with η) or be multiplied by 0. The latter is the case if $p_I(y) > 0$ but $\mu(x, y) \neq z$. The number of times in which a non-zero value $p_I(y)$ gets set to zero can only be finite. Thus, there are only finitely many distributions p_I that have to be considered. Notice that our argument exploits the fact that for deterministic hidden state, the number of non-zero p_I-value never increases.

We note, however, that the space of information states is generally much larger than the state space itself. If, for example, the number of hidden states y is m, the number of information states that can be defined over y is $\Omega(2^m)$—although not all of them might plausibly occur. Nevertheless, any algorithm for computing a policy for action selection is likely to be dominated by the hidden state y, rather than the observable state x. In the next section, we will exploit this insight by using state space search to develop performance bounds for the much harder problem of finding solutions in information state space.

3 BA* Search

This section describes a search algorithm proposed for solving MDPDHSs which we will refer to as *BA** (short for *belief space A**). Our algorithm only applies to MDPDHSs that terminate, e.g., by reaching a goal state. At the core, our search algorithm is similar to the decades-old A* algorithm [9].

A primary difference is that BA* searches in the space of all information states, instead of in the space of all states. Just like many other modern-day implementations of heuristic search [12], our approach uses memoization with a hashing function to avoid searching the same information state twice—thereby reaping the same benefits as, for example, value iteration does.

The key algorithmic innovation in this paper is a method for generating bounds that are utilized to prune the search tree and as a search heuristic when deciding what node to expand next. As we will demonstrate in our experiments, this mechanism can reduce the computation for finding a policy by several orders of magnitude.

3.1 Search in Information Space

At the core of the BA* search algorithm is a search tree, similar but not identical to trees found in the stochastic games literature [1]. The tree possesses two types of nodes: *choice nodes* and *response nodes* (see also [5]). Each of these nodes is labeled with an information state $\langle x; p_I \rangle$. Choice nodes represent situations where the agent can make an action choice. Since the agent maximizes its reward, it will pick the action u that maximizes the value of its children. Response nodes react to the action choice in a non-deterministic way. The non-determinism arises from two factors: The stochastic nature of the MDP, and the unknown hidden state y. Thus, the search algorithm averages the values of the response nodes, weighted in proportion to the probability of each succeeding information state.

Mathematically, the tree is used to recursively calculate an expected cumulative reward for each choice node in the tree. Let $V(x, p_I)$ denote the expected cumulative reward for being in the information state $\langle x; p_I \rangle$. $V(x, p_I)$ is defined recursively through the value of its children and their children:

$$V(x,p_I) \quad \longleftarrow \quad R(x) + \gamma \max_u \sum_y p_I(y) \sum_{x'} p(x'|x,y,u) V(x', p_I') \tag{4}$$

where p_I' is the distribution over states y (i.e., the information state with regards to the hidden state y) that results by applying action u to the state $\langle x; y \rangle$ and the initial information state p_I. The equations for arriving at p_I' are given by (2) and (3). Thus, the search tree is a max-avg tree in information space, where choice nodes maximize the V-function, and response nodes average it.

3.2 Propagating Bounds

The key innovation of the BA* search algorithm is a mechanism for calculating bounds on the V-value of each information state. These bounds are used to guide the heuristic search (which node to expand next) and they form also the basis for deciding which branches to prune entirely in the search. They are the key reason for the computational efficiency of our approach, which will be documented extensively in the experimental results section of this paper.

The intuitive idea is simple: Suppose we would like to know the V-value of an information state $\langle x; p_I \rangle$. Instead of performing the full search in information space, we can search the observable state space x instead. This search is exponentially faster, since the state space is exponentially smaller than the information space (in most cases). To search in state space, we need to assume knowledge of the hidden state y. This raises the question as to what hidden state y shall we use in the search. The answer is straightforward: If at any point in time, we choose the hidden state that maximally beneficial to our goal of maximizing reward, the resulting value function will be an upper bound to the true V-value. Similarly, if we always assume the worst state y, the result will be a lower bound.

We will now formalize this idea. Let α be the desired lower bound, and β the desired upper bound. Then the bounds are defined recursively as follows:

$$\begin{aligned}
\alpha(x,p_I) \quad &\longleftarrow \quad R(x) + \gamma \max_u \min_{y:p_I(y)>0} \sum_{x'} p(x'|x,y,u)\alpha(x', p_I) \\
\beta(x,p_I) \quad &\longleftarrow \quad R(x) + \gamma \max_u \max_{y:p_I(y)>0} \sum_{x'} p(x'|x,y,u)\beta(x', p_I)
\end{aligned} \tag{5}$$

where p_I is the present information state regarding the hidden state y. It is straightforward to show these values bound the V-function:

$$\alpha(x,p_I) \quad \leq \quad V(x,p_I) \quad \leq \quad \beta(x,p_I) \tag{6}$$

Notice that for any fixed p_I, these bounds are calculated entirely in the state space of the observable state x, and not in the hidden state y or the more complex information state space. Thus, they can be found very efficiently. In our implementation, they are calculated using value iteration, and their calculation consumes a negligible fraction of the overall search time.

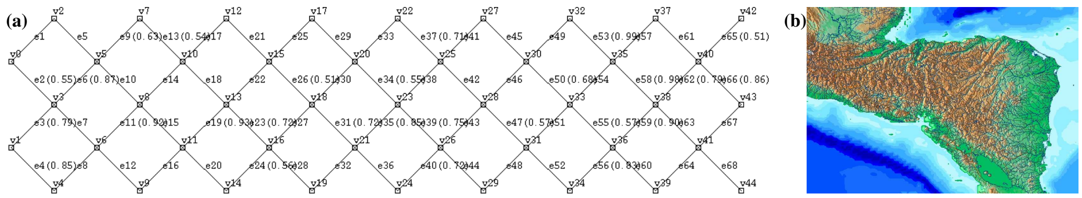

Figure 1: (a) Test graph for systematic experiments. (b) Map of Honduras, used for large scale experiments.

We also notice that these bounds are different from the familiar alpha beta bounds in two player game search. Those bounds are only applicable to min-max search. Our bounds exploit the duality between information space and state space, and they are calculated using a separate search algorithm. However, we conjecture that they are just as useful in searching MDPDHSs as the alpha beta bounds are in solving two-player games.

3.3 Pruning and Search Heuristic

The bounds play a key role in pruning the search tree, and in determining which node to expand next.

- **Pruning.** Consider a choice node. Here we choose the action u that *maximizes* the expected V-value of its sub-tree. Clearly, if the β-value of a sub-tree is smaller than the α-value of another, it is impossible that this subtree corresponds to the optimal action and hence it can be pruned. As we will see below, pruning trees in this way can dramatically reduce the overall computation.
- **Search heuristic.** If more than one branch remains in a choice node after pruning, the α-value is used to heuristically determine which node to search next. This heuristic is used in a way very similar to A*. The only difference here is that the priority of an information state factors in the probability that this information state will be reached, which is easily obtained by following the path to the information state in the search tree. In the language of A*, this function constitutes an *admissible heuristic* in that it underestimates the true value of information states [9]. As a result, our algorithm is guaranteed to terminate in goal-seeking tasks, a property commonly known as *completeness*. Our approach differs from A* in that the heuristic is not supplied by a human designer. Instead, it is derived automatically from the problem description.

This completes the description of the BA* search algorithm. In summary, BA* performs search in the space of all information states. The key insight is that bounds on the expected value of an information state can quickly be obtained by state-space search, which in general will be exponentially faster than the search in information space. Utilizing this bound, the search tree can be pruned. The bounds also provide a search heuristic for determining which node to expand next.

4 Experimental Results

Systematic experimental results were performed to evaluate the scaling properties of our algorithm in domains with hidden state. In particular, we compared our algorithm to the following two algorithms: an optimized Q-learning algorithm [16], which we consider a state-of-the-art MDP solution, and Cassandra's et al. *POMDP solver software*[1] for computing policies in POMDPs [5]. Additional experimental results illustrate the effect of pruning and heuristic search using the bounds calculated via the approximate MDPs. Across the board, we found that our solution outperforms alternative approaches by a large margin. In certain experiments, our approach appears to scale linearly with the domain size, whereas the alternatives scale exponentially. We also find that the bounds are essential to keep computation low, especially in large state spaces.

In detail, the experimental world is shown in Figure 1a. The problem is to move from the left end of the world to the right end. Arcs in the graph might or might not be traversable, but the only way to find out is to move there and try. The problem is to find an optimal contingency plan that enables the agent to recover is an arc is found to be impassable. This problem is known as the NP-hard *Canadian Travelers Problem* [10], and is very similar to the mobile robot exploration problem [13]. The advantage of an artificial environment such as the one used here is that it complexity can easily be varied, by adding new nodes to the graph. The number of arcs, and hence the number of hidden states, is linear in the number

[1] See http://www.cassandra.org/pomdp/code/

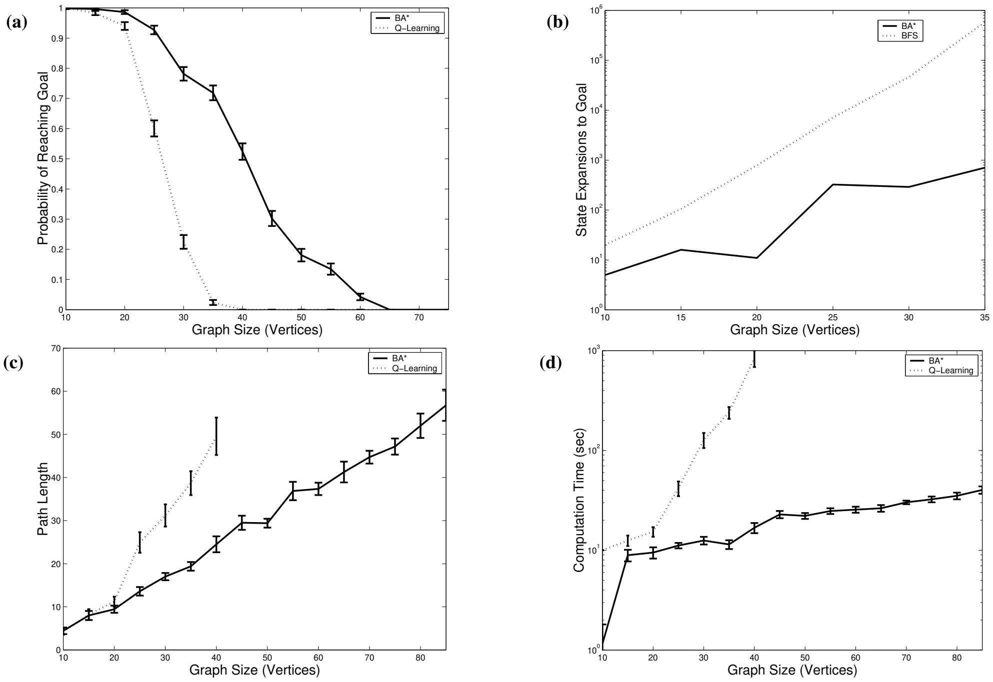

Figure 2: Performance curves, comparing Q-learning with explicit encoding of the hidden state with our new search algorithm. In all graphs, the horizontal axis depicts the number of nodes in the world. (a) Success rate for a single policy calculated up front. (b) Number of nodes expanded by our approach versus breadth first search. (c) Average length of the path to the goal, using replanning. (d) Average computation time required to find a policy of fixed quality.

of nodes in the graph. As a direct consequence, the belief state space is exponential in the size of the state space.

The *POMDP solver software* was only able to generate policies in extremely small worlds. This is not surprising, as POMDPs are more general than MDPDHSs. The timing results are as follows, broken down into the time required for finding an initial policy and for finding the optimal policy:

number of nodes in graph	initial solution	optimal solution
4	5 sec	$\sim$5,000 sec
5	8 sec	$\sim$7,000 sec
8	$>$ 3,000 sec	$>$ 24hrs

We therefore did not investigate this approach any further.

Due to the restrictive nature of MDPDHSs, the MDP solution scales much better. As stated above, Q-learning was used on the discrete information state space. Figure 2a compares the performance of Q-Learning (dashed line) and our search algorithm (solid line) under equal computational conditions. Shown there is the probability of reaching the goal with a policy calculated with fixed computational time. As is easy to be seen, our approach can handle significantly larger worlds. Notice that the horizontal axis depicts the size of the graph. The size of the information space is much larger. These results reflect the performance if the policy is calculated up front and never revised as the agent moves.

A common strategy for boosting the performance in worlds with hidden state is to replan whenever new information arrives. Figure 2c&d show performance results for such a setting. In particular, Figure 2c shows the average path length of the controller identified by Q-learning (dashed curve) and compares it with the average path length obtained using our heuristic search algorithm. Notice that our approach scales significantly better. From worlds with 45 states on, Q-learning fails to find any policy that even contains a single path to the goal in reasonable time, despite replanning. This is because the initial planning step never finds even a single path to the goal. Our approach navigates successfully in much larger worlds. Figure 2d shows related results: Plotted here is the computation time required to achieve a fixed solution quality. Again, the Q-learning implementation scales much poorer to complex worlds than our new heuristic search algorithm.

These results raise the question as to *why* the search algorithm is so much better than Q-learning. We conjecture that our approach for generating bounds and using them to prune and order the search makes the difference. Figure 2b illustrates the effect of the bounds by plotting the number information states expanded in the search, for our algorithm and breadth-first search (which is our algorithm deprived of the bounds). Notice that the vertical axis uses a logarithmic scale. Obviously, our bounds reduce the number of nodes expanded by more than three orders of magnitudes, and the gap increases with the size of the world. We view this as a key result for motivating the MDPDHS model—instead of subsuming the type problems addressed in this paper into the more common MDP model. It is the specific hidden state structure that makes our pruning and heuristic search methods possible.

We also applied our algorithm to a large-scale navigation problem using data of Honduras released by the USGS (see Figure 1b), as part of a DARPA-sponsored research program on disaster relief. The specific scenario involved roads randomly destroyed by Hurricane Mitch, which created a fancy version of the Canadian Travelers Problem. Our map contained 234 nodes and 300 major road segments, creating an information state space of size $4.7 \cdot 10^{90}$ states. Using replanning, our approach consistently managed to move from any location to any other on a convincingly short path, assuming that such a path exists in the first place. We are not aware of any other algorithm that could handle domains of this size, but we also cannot assess how close to optimal our approach performs.

5 Related Work

To best of our knowledge, the framework of MDPDHS has not been studied before. The work presented here is immanently related to a rich body of literature on problem solving and learning. The relation to MDPs and POMDPs was already discussed above. While MDPs lack a notion of hidden state, the generality of POMDPs make them presently inapplicable to all but the simplest problems. The MDPDHS model lies in between, inheriting some of the computational efficiency from the MDP model, while being able to cope with certain types of hidden state. The work is also related to the rich literature on learning finite state machines [11, 7]. Like ours, the problem addressed in this literature is characterized by deterministic hidden state. It differs, however, in the control goal (system identification versus reward maximization), and it does not incorporate a model of control noise of any type, in contrast to the problem studied here. Similarly, previous research on the Canadian Travelers Problem has also not considered the issue of control noise [10].

6 Discussion

We have presented an efficient algorithm for finding policies in MDPs with deterministic hidden states (MDPDHS). Our algorithm interleaves search in information state space (old idea) with optimistic/pessimistic search in state space to derive performance bounds (new idea). These bounds are used for pruning the search tree and for determining which node to expand next. Extensive experimental comparisons show that our approach outperforms state-of-the-art MDP and POMDP solutions to this problem.

As stated in the introduction to this paper, we believe that MDPs and POMDPs are only the two ends of a spectrum of sequential decision making problems under uncertainty. We hope that this research motivates future research on extensions of MDPs that capture some relevant aspects of POMDPs, yet lend themselves to computationally more efficient solutions than the full-blown POMDP solution.

References

[1] M. Bardi, Parthasarathym T., and T.E.S. Raghavan. *Stochastic and Differential Games: Theory and Numerical Methods*. Birkhauser, 1999.

[2] J. Baxter, L. Weaver, and P. Bartlett. Infinite-horizon gradient-based policy search: Ii. gradient ascent algorithms and experiments. *JAIR*, to appear.

[3] C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic, high-level robot programming in the situation calculus. *AAAI-2000*.

[4] A.M. Jazwinsky. *Stochastic Processes and Filtering Theory*. Academic, 1970.

[5] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2), 1998.

[6] S. Koenig. The complexity of real-time search. TR CMU-CS-92-145, Carnegie Mellon Univ.

[7] M. C. Mozer and J. R. Bachrach. Discovering the structure of a reactive environment by exploration. TR CU-CS-451-89, Univ. of Colorado, 1989.

[8] A.Y. Ng and M. Jordan. PEGASUS: a policy search method for large MDPs and POMDPs. *UAI-2000*.

[9] N. J. Nilsson. *Principles of Artificial Intelligence*. Springer, 1982.

[10] C. Papadimitriou and M. Yannakakis. Shortest paths without a map. *ICALP-89*.

[11] R. L. Rivest and R. E. Schapire. Diversity-based inference of finite automata. In *FOCS-87*.

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

[13] R. Simmons at al. Coordination for multi-robot exploration and mapping. *AAAI-2000*.

[14] S. Singh, M. Kearns, D. Litman, and M. Walker. Reinforcement learning for spoken dialogue systems. *NIPS-11*, 2000.

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

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

[17] R. J. Williams and L. C. Baird III. Analysis of some incremental variant of policy iteration: First steps toward understanding actor-critic learning systems. TR NU-CCS-93-11, Northeastern Univ., 1993.