Journal of Machine Learning Research 12 (2011) 691-730 Submitted 2/10; Revised 11/10; Published 3/11

Inverse Reinforcement Learning in Partially Observable Environments

Jaedeug Choi JDCHOI@AI.KAIST.AC.KR
Kee-Eung Kim KEKIM@CS.KAIST.AC.KR
Department of Computer Science
Korea Advanced Institute of Science and Technology
Daejeon 305-701, Korea

Editor: Shie Mannor

Abstract

Inverse reinforcement learning (IRL) is the problem of recovering the underlying reward function from the behavior of an expert. Most of the existing IRL algorithms assume that the environment is modeled as a Markov decision process (MDP), although it is desirable to handle partially observable settings in order to handle more realistic scenarios. In this paper, we present IRL algorithms for partially observable environments that can be modeled as a partially observable Markov decision process (POMDP). We deal with two cases according to the representation of the given expert's behavior, namely the case in which the expert's policy is explicitly given, and the case in which the expert's trajectories are available instead. The IRL in POMDPs poses a greater challenge than in MDPs since it is not only ill-posed due to the nature of IRL, but also computationally intractable due to the hardness in solving POMDPs. To overcome these obstacles, we present algorithms that exploit some of the classical results from the POMDP literature. Experimental results on several benchmark POMDP domains show that our work is useful for partially observable settings.

Keywords: inverse reinforcement learning, partially observable Markov decision process, inverse optimization, linear programming, quadratically constrained programming

1. Introduction

Inverse reinforcement learning (IRL) was first proposed by Russell (1998) as follows:

Given (1) measurements of an agent's behavior over time, in a variety of circumstances, (2) measurements of the sensory inputs to the agent, (3) a model of the physical environment (including the agent's body).

Determine the reward function that the agent is optimizing.

The significance of IRL has emerged from the connection between reinforcement learning (RL) and other research areas such as neurophysiology (Montague and Berns, 2002; Cohen and Ranganath, 2007), behavioral neuroscience (Lee et al., 2004; Niv, 2009) and economics (Erev and Roth, 1998; Borgers and Sarin, 2000; Hopkins, 2007). In these research areas, the reward function is generally assumed to be fixed and known, but it is often non-trivial to come up with an appropriate reward function for each problem. Hence, a progress in IRL can have a significant impact on many research areas.

IRL is a natural way to examine animal and human behaviors. If the decision maker is assumed to follow the principle of rationality (Newell, 1982), its behavior could be understood by the reward function that the decision maker internally optimizes. In addition, we can exploit the computed reward function to generate an agent that imitates the decision maker's behavior. This will be a useful approach to build an intelligent agent. Another advantage of IRL is that the solution of IRL problems, that is, the reward function, is one of the most transferable representations of the agent's behavior. Although it is not easy to transfer the control policy of the agent to other problems that have a similar structure with the original problem, the reward function could be applied since it compactly represents the agent's objectives and preferences.

In the last decade, a number of studies on IRL have been reported. However, most of the previous IRL algorithms (Ng and Russell, 2000; Abbeel and Ng, 2004; Ramachandran and Amir, 2007; Neu and Szepesvari, 2007; Syed and Schapire, 2008; Ziebart et al., 2008) assume that the agent acts in an environment that can be modeled as a Markov decision process (MDP). Although the MDP assumption provides a good starting point for developing IRL algorithms, the implication is that the agent has access to the true global state of the environment. The assumption of an omniscient agent is often too strong in practice. Even though the agent is assumed to be an expert in the given environment, the agent may be (and often is) making optimal behaviors with a limited sensory capability. Hence, to relax the strong assumption and widen the applicability of IRL to more realistic scenarios, the IRL algorithms should be extended to partially observable environments, which can be modeled as partially observable Markov decision processes.

A partially observable Markov decision process (POMDP) (Sondik, 1971; Monahan, 1982; Kaelbling et al., 1998) is a general mathematical framework for single-agent planning under uncertainty about the effect of actions and the true state of the environment. Recently, many approximate techniques have been developed to compute an optimal control policy for large POMDPs. Thus, POMDPs have increasingly received a significant amount of attention in diverse research areas such as robot navigation (Spaan and Vlassis, 2004; Smith, 2007), dialogue management (Williams and Young, 2007), assisted daily living (Hoey et al., 2007), cognitive radio (Zhao et al., 2007) and network intrusion detection (Lane and Brodley, 2003). However, in order to address real-world problems using POMDPs, first, a model of the environment and the reward function should be obtained. The parameters for the model of an environment, such as transition probabilities and observation probabilities, can be computed relatively easily by counting the events if the true state can be accessed, but determining the reward function is non-trivial. In practice, the reward function is repeatedly hand-tuned by domain experts until a satisfactory policy is acquired. This usually entails a labor intensive process. For example, when developing a spoken dialogue management system, POMDP is a popular framework for computing the dialogue strategy, since we can compute an optimal POMDP policy that is robust to speech recognition error and maintains multiple hypotheses of the user's intention (Williams and Young, 2007). In this domain, transition probabilities and observation probabilities can be calculated from the dialogue corpus collected from a wizard-of-oz study. However, there is no straightforward way to compute the reward function, which should represent the balance among the reward of a successful dialogue, the penalty of an unsuccessful dialogue, and the cost of information gathering. It is manually adjusted until a satisfying dialogue policy is obtained. Therefore, a systematic method is desired to determine the reward function.

In this paper, we describe IRL algorithms for partially observable environments extending our previous results in Choi and Kim (2009). Specifically, we assume that the environment is modeled as a POMDP and try to compute the reward function given that the agent follows an optimal policy.

The algorithm is mainly motivated by the classical IRL algorithm by Ng and Russell (2000) and we adapt the algorithm to be robust for large problems by using the methods suggested by Abbeel and Ng (2004). We believe that some of the more recently proposed IRL algorithms (Ramachandran and Amir, 2007; Neu and Szepesvari, 2007; Syed and Schapire, 2008; Ziebart et al., 2008) also could be extended to handle partially observable environments. The aim of this paper is to present a general framework for dealing with partially observable environments, the computational challenges involved in doing so, and some approximation techniques for coping with the challenges. Also, we believe that our work will prove useful for many problems that could be modeled as POMDPs.

The remainder of the paper is structured as follows: Section 2 reviews some definitions and notations of MDP and POMDP. Section 3 presents an overview of the IRL algorithms by Ng and Russell (2000) and Abbeel and Ng (2004). Section 4 gives a formal definition of IRL for partially observable environments, and discusses the fundamental difficulties of IRL and the barriers of extending IRL to partially observable environments. In Section 5, we focus on the problem of IRL with the explicitly given expert's policy. We present the optimality conditions of the reward function and the optimization problems with the computational challenges and some approximation techniques. Section 6 deals with more practical cases where the trajectories of the expert's actions and observations are given. We present algorithms that iteratively find the reward function, comparing the expert's policy and other policies found by the algorithm. Section 7 shows the experimental results of our algorithms in several POMDP domains. Section 8 briefly reviews related work on IRL. Finally, Section 9 discusses some directions for future work.

2. Preliminaries

Before we present the IRL algorithms, we briefly review some definitions and notations of MDP and POMDP to formally describe the completely observable environment and the partially observable environment.

2.1 Markov Decision Process

A Markov decision process (MDP) provides a mathematical framework for modeling a sequential decision making problem under uncertainty about the effect of an agent's action in an environment where the current state depends only on the previous state and action, namely, the Markov property. An MDP is defined as a tuple $\langle S, A, T, R, \gamma \rangle$:

- S is the finite set of states.
- A is the finite set of actions.
- $T : S \times A \to \Pi(S)$ is the state transition function, where $T(s,a,s')$ denotes the probability $P(s'|s,a)$ of reaching state s' from state s by taking action a.
- $R : S \times A \to \mathbb{R}$ is the reward function, where $R(s,a)$ denotes the immediate reward of executing action a in state s, whose absolute value is bounded by R_{max}.
- $\gamma \in [0,1)$ is the discount factor.

A policy in MDP is defined as a mapping $\pi : S \to A$, where $\pi(s) = a$ denotes that action a is always executed in state s following the policy π. The value function of policy π at state s is the

expected discounted return of starting in state s and executing the policy. The value function can be computed as:

$$V^{\pi}(s) = R(s,\pi(s)) + \gamma \sum_{s' \in S} T(s,\pi(s),s') V^{\pi}(s'). \tag{1}$$

Given an MDP, the agent's objective is to find an optimal policy π^* that maximizes the value for all the states, which should satisfy the Bellman optimality equation:

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

It is often useful to express the above equation in terms of Q-function: π is an optimal policy if and only if

$$\pi(s) \in \operatorname*{argmax}_{a \in A} Q^{\pi}(s,a),$$

where

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^{\pi}(s'), \tag{2}$$

which is the expected discounted return of executing action a in state s and then following the policy π.

2.2 Partially Observable Markov Decision Process

A partially observable Markov decision process (POMDP) is a general framework for modeling the sequential interaction between an agent and a partially observable environment where the agent cannot completely perceive the underlying state but must infer the state based on the given noisy observation. A POMDP is defined as a tuple $\langle S, A, Z, T, O, R, b_0, \gamma \rangle$:

- S, A, T, R and γ are defined in the same manner as in MDPs.
- Z is the finite set of observations.
- $O : S \times A \to \Pi(Z)$ is the observation function, where $O(s,a,z)$ denotes the probability $P(z|s,a)$ of perceiving observation z when taking action a and arriving in state s.
- b_0 is the initial state distribution, where $b_0(s)$ denotes the probability of starting in state s.

Since the true state is hidden, the agent has to act based on the history of executed actions and perceived observations. Denoting the set of all possible histories at the t-th time step as $H_t = (A \times Z)^t$, a policy in POMDP is defined as a mapping from histories to actions $\pi : H_t \to A$. However, since the number of possible histories grows exponentially with the number of time steps, many POMDP algorithms use the concept of *belief*. Formally, the belief b is the probability distribution over the current states, where $b(s)$ denotes the probability that the state is s at the current time step, and Δ denotes a $|S|-1$ dimensional *belief simplex*. The belief update for the next time step can be computed from the belief at the current time step: Given the action a at the current time step and the observation z at the next time step, the updated belief b_z^a for the next time step is obtained by

$$b_z^a(s') = P(s'|b,a,z) = \frac{O(s',a,z) \sum_s T(s,a,s') b(s)}{P(z|b,a)}, \tag{3}$$

where the normalizing factor $P(z|b,a) = \sum_{s'} O(s',a,z) \sum_s T(s,a,s')b(s)$. Hence, the belief serves as a sufficient statistic for fully summarizing histories, and the policy can be equivalently defined as a mapping $\pi : \Delta \to A$, where $\pi(b) = a$ specifies action a to be selected at the current belief b by the policy π. Using beliefs, we can view POMDPs as belief-state MDPs, and the value function of an optimal policy satisfies the Bellman equation:

$$V^*(b) = \max_a \Big[\sum_s b(s)R(s,a) + \gamma \sum_{s',z} T(s,a,s')O(s',a,z)V^*(b_z^a) \Big]. \tag{4}$$

Alternatively, a policy in POMDP can be represented as a finite state controller (FSC). An FSC policy is defined by a directed graph $\langle \mathcal{N}, \mathcal{E} \rangle$, where each node $n \in \mathcal{N}$ is associated with an action $a \in A$ and has an outgoing edge $e_z \in \mathcal{E}$ per observation $z \in Z$. The policy can be regarded as $\pi = \langle \psi, \eta \rangle$ where ψ is the *action strategy* associating each node n with an action $\psi(n) \in A$, and η is the *observation strategy* associating each node n and observation z with a successor node $\eta(n,z) \in \mathcal{N}$.

Given an FSC policy $\pi = \langle \psi, \eta \rangle$, the value function V^π is the expected discounted return of executing π and is defined over the joint space of FSC nodes and POMDP states. It can be computed by solving a system of linear equations:

$$V^\pi(n,s) = R(s,a) + \gamma \sum_{n',s'} T^{a,os}(\langle n,s \rangle, \langle n',s' \rangle) V^\pi(n',s'), \tag{5}$$

where

$$T^{a,os}(\langle n,s \rangle, \langle n',s' \rangle) = T(s,a,s') \sum_{\substack{z \in Z \text{ s.t.} \\ os(z) = n'}} O(s',a,z), \tag{6}$$

with $a = \psi(n)$ and $os(z) = \eta(n,z)$. The value at node n for belief b is calculated by

$$V^\pi(n,b) = \sum_s b(s) V^\pi(n,s), \tag{7}$$

and the starting node for the initial belief b_0 is chosen by $n_0 = \text{argmax}_n V^\pi(n,b_0)$. We can also define Q-function for an FSC policy π:

$$Q^\pi(\langle n,s \rangle, \langle a,os \rangle) = R(s,a) + \gamma \sum_{n',s'} T^{a,os}(\langle n,s \rangle, \langle n',s' \rangle) V^\pi(n',s'),$$

which is the expected discounted return of choosing action a at node n and moving to node $os(z)$ upon observation z, and then following policy π. Also, Q-function for node n at belief b is computed by

$$Q^\pi(\langle n,b \rangle, \langle a,os \rangle) = \sum_s b(s) Q^\pi(\langle n,s \rangle, \langle a,os \rangle).$$

With an FSC policy π, we can sort the reachable beliefs into nodes, such that B_n denotes the set of beliefs that are reachable from the initial belief b_0 and the starting node n_0 when the current node is n. Note that $|B_n| \geq 1$ for every node n.

3. IRL in Completely Observable Markovian Environments

The MDP framework provides a good starting point for developing IRL algorithms in completely observable Markovian environments and most of the previous IRL algorithms address the problems in the MDP framework. In this section, we overview the IRL algorithms proposed by Ng and Russell (2000) and Abbeel and Ng (2004) as background to our work.

The IRL problem in completely observable Markovian environments is denoted with IRL for MDP\R, which is formally stated as follows: Given an MDP\R $\langle S, A, T, \gamma \rangle$ and an expert's policy π_E, find the reward function R that makes π_E an optimal policy for the given MDP. The problem can be categorized into two cases: The first case is when an expert's policy is explicitly given and the second case is when an expert's policy is implicitly given by its trajectories.

3.1 IRL for MDP\R from Policies

Let us assume that an expert's policy π_E is explicitly given. Ng and Russell (2000) present a necessary and sufficient condition for the reward function R of an MDP to guarantee the optimality of π_E:

$$Q^{\pi_E}(s, \pi_E(s)) \geq Q^{\pi_E}(s, a), \quad \forall s \in S, \forall a \in A, \tag{8}$$

which states that deviating from the expert's policy should not yield a higher value. From the condition, they suggest the following:

Theorem 1 [Ng and Russell, 2000] *Let an MDP\R $\langle S, A, T, \gamma \rangle$ be given. Then the policy π is optimal if and only if the reward function R satisfies*

$$R^{\pi} - R^{a} + \gamma(T^{\pi} - T^{a})(I - \gamma T^{\pi})^{-1} R^{\pi} \succeq 0, \quad \forall a \in A, \tag{9}$$

where the matrix notations and the matrix operator are defined as follows:

- *T^{π} is a $|S| \times |S|$ matrix with (s, s') element being $T(s, \pi(s), s')$.*
- *T^{a} is a $|S| \times |S|$ matrix with (s, s') element being $T(s, a, s')$.*
- *R^{π} is a $|S|$ vector with s-th element being $R(s, \pi(s))$.*
- *R^{a} is a $|S|$ vector with s-th element being $R(s, a)$.*
- *V^{π} is a $|S|$ vector with s-th element being $V^{\pi}(s)$.*
- *$X \succeq Y \Leftrightarrow X(i) \geq Y(i)$, for all i, if the length of X is the same as that of Y.*

Proof Equation (1) can be rewritten as $V^{\pi} = R^{\pi} + \gamma T^{\pi} V^{\pi}$. Thus,

$$V^{\pi} = (I - \gamma T^{\pi})^{-1} R^{\pi}. \tag{10}$$

By the definition of an optimal policy and Equation (2), π is optimal if and only if

$$\begin{aligned}
\pi(s) &\in \operatorname*{argmax}_{a \in A} Q^{\pi}(s, a), \quad \forall s \in S \\
&= \operatorname*{argmax}_{a \in A} (R(s, a) + \gamma \sum_{s'} T(s, a, s') V^{\pi}(s')), \quad \forall s \in S \\
\Leftrightarrow R(s, \pi(s)) &+ \gamma \sum_{s'} T(s, \pi(s), s') V^{\pi}(s') \\
&\geq R(s, a) + \gamma \sum_{s'} T(s, a, s') V^{\pi}(s'), \quad \forall s \in S, \forall a \in A.
\end{aligned}$$

$$\begin{aligned}
\text{maximize}_R \quad & \sum_s \sum_{a \in A \setminus \pi_E(s)} \left[Q^{\pi_E}(s, \pi_E(s)) - Q^{\pi_E}(s,a) \right] - \lambda \|R\|_1 \\
\text{subject to} \quad & R^{\pi_E} - R^a + \gamma (T^{\pi_E} - T^a)(I - \gamma T^{\pi_E})^{-1} R^{\pi_E} \succeq 0, \quad \forall a \in A \\
& |R(s,a)| \leq R_{max}, \quad \forall s \in S, \forall a \in A
\end{aligned}$$

Table 1: Optimization problem of IRL for MDP\R from the expert's policy.

By rephrasing with the matrix notations and substituting with Equation (10),

$$\begin{aligned}
& R^\pi + \gamma T^\pi V^\pi \succeq R^a + \gamma T^a V^\pi, \quad \forall a \in A \\
\Leftrightarrow\ & R^\pi + \gamma T^\pi (I - \gamma T^\pi)^{-1} R^\pi \succeq R^a + \gamma T^a (I - \gamma T^\pi)^{-1} R^\pi, \quad \forall a \in A \\
\Leftrightarrow\ & R^\pi - R^a + \gamma (T^\pi - T^a)(I - \gamma T^\pi)^{-1} R^\pi \succeq 0, \quad \forall a \in A.
\end{aligned}$$

■

Equation (9) bounds the feasible space of the reward functions that guarantee the optimality of the expert's policy, and there exist infinitely many reward functions that satisfy Equation (9). As a degenerate case, $R = 0$ is always a solution. Thus, given the expert's policy π_E, which is assumed to be optimal, the reward function is found by solving the optimization problem in Table 1, where λ is an adjustable weight for the penalty of having too many non-zero entries in the reward function. The objective is to maximize the sum of the margins[1] between the expert's policy and all other policies that deviate a single step from the expert's policy, in the hope that the expert's policy is optimal while favoring sparseness in the reward function.

3.2 IRL for MDP\R from Sampled Trajectories

In some cases, we have to assume that the expert's policy is not explicitly given but instead the trajectories of the expert's policy in the state and action spaces are available.[2] The m-th trajectory of the expert's policy is defined as the H-step state and action sequences $\{s_0^m, s_1^m, \cdots, s_{H-1}^m\}$ and $\{a_0^m, a_1^m, \cdots, a_{H-1}^m\}$.

In order to address problems with large state spaces, Ng and Russell (2000) use a linear approximation for the reward function, and we also assume that the reward function is linearly parameterized as

$$R(s,a) = \alpha_1 \phi_1(s,a) + \alpha_2 \phi_2(s,a) + \cdots + \alpha_d \phi_d(s,a) = \alpha^T \phi(s,a), \tag{11}$$

where known basis functions $\phi : S \times A \to [0,1]^d$ and the weight vector $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_d]^T \in \mathbb{R}^d$. We also assume without loss of generality that $\alpha \in [-1,1]^d$.

1. We found it more successful to use the sum-of-margins approach than the minimum-of-margins approach in the original paper, since the latter may fail when there are multiple optimal policies.
2. Although only the trajectories of states and actions are available, the transition function T is assumed to be known in MDP\R.

Algorithm 1 IRL for MDP\R from the sampled trajectories using LP.

Input: MDP\R $\langle S, A, T, \gamma \rangle$, basis functions ϕ, M trajectories

1: Choose a random initial policy π_1 and set $\Pi = \{\pi_1\}$.
2: **for** $k = 1$ to $MaxIter$ **do**
3: Find $\hat{\alpha}$ by solving the linear program:

$$\text{maximize}_{\hat{\alpha}} \quad \sum_{\pi \in \Pi} p\left(\hat{V}^{\pi_E}(s_0) - \hat{V}^{\pi}(s_0)\right)$$
$$\text{subject to} \quad |\hat{\alpha}_i| \leq 1 \quad i = 1, 2, \cdots, d$$

4: Compute an optimal policy π_{k+1} for the MDP with $\hat{R} = \hat{\alpha}^T \phi$.
5: **if** $\hat{V}^{\pi_E}(s_0) - V^{\pi_{k+1}}(s_0) \leq \varepsilon$ **then**
6: **return** $\hat{R}$
7: **else**
8: $\Pi = \Pi \cup \{\pi_{k+1}\}$
9: **end if**
10: **end for**
11: **return** $\hat{R}$

Output: the reward function $\hat{R}$

Then, from the given M trajectories, the value of π_E for the starting state s_0 is estimated by the average empirical return for an estimated reward function $\hat{R} = \hat{\alpha}^T \phi$:

$$\hat{V}^{\pi_E}(s_0) = \frac{1}{M} \sum_{m=1}^{M} \sum_{t=0}^{H-1} \gamma^t \hat{R}(s_t^m, a_t^m) = \frac{1}{M} \hat{\alpha}^T \sum_{m=1}^{M} \sum_{t=0}^{H-1} \gamma^t \phi(s_t^m, a_t^m).$$

The algorithm is presented in Algorithm 1. It starts with the set of policies Π initialized by a *base case* random policy π_1. Ideally, the true reward function R should yield $V^{\pi_E}(s_0) \geq V^{\pi}(s_0)$ for $\forall \pi \in \Pi$ since the expert's policy π_E is assumed to be an optimal policy with respect to R. The values of other policies with a candidate reward function $\hat{R}$ are either estimated by sampling trajectories or are exactly computed by solving the Bellman equation, Equation (1). The algorithm iteratively tries to find a better reward function $\hat{R}$, given the set of policies Π found by the algorithm $\Pi = \{\pi_1, \ldots, \pi_k\}$ up to iteration k, by solving the optimization problem in line 3, where $p(x)$ is a function that favors $x > 0$.[3] The algorithm then computes a new policy π_{k+1} that maximizes the value function under the new reward function, and adds π_{k+1} to Π. The algorithm continues until it has found a satisfactory reward function.

The above algorithm was extended for the apprenticeship learning in the MDP framework by Abbeel and Ng (2004). The goal of apprenticeship learning is to learn a policy from an expert's demonstrations without a reward function, so it does not compute the exact reward function that the expert is optimizing but rather the policy whose performance is close to that of the expert's policy on the unknown reward function. This is worth reviewing, as we adapt this algorithm to address the IRL problems in partially observable environments.

We assume that there are some known basis functions ϕ and the reward function is linearly parameterized with the weight vector α as in Equation (11). Also, assume $\|\alpha\|_1 \leq 1$ to bound R_{max} by 1. The value of a policy π can be written using the feature expectation $\mu(\pi)$ for the reward

3. Ng and Russell (2000) chose $p(x) = x$ if $x \geq 0$, and $p(x) = 2x$ if $x < 0$ in order to favor $x > 0$ but more penalize $x < 0$. The coefficient of 2 was heuristically chosen.

Algorithm 2 Apprenticeship learning using QCP.

Input: MDP\R $\langle S, A, T, \gamma \rangle$, basis functions ϕ, M trajectories

1: Choose a random initial weight α and set $\Pi = \emptyset$.
2: **repeat**
3: Compute an optimal policy π for the MDP with $R = \alpha^T \phi$.
4: $\Pi = \Pi \cup \{\pi\}$
5: Solve the following optimization problem:

$$\begin{aligned} &\text{maximize}_{t,\alpha} \quad t \\ &\text{subject to} \quad \alpha^T \mu_E \geq \alpha^T \mu(\pi) + t, \quad \forall \pi \in \Pi \\ &\qquad\qquad \|\alpha\|_2 \leq 1 \end{aligned}$$

6: **until** $t \leq \varepsilon$

Output: the reward function R

Algorithm 3 Apprenticeship learning using the projection method.

Input: MDP\R $\langle S, A, T, \gamma \rangle$, basis functions ϕ, M trajectories

1: Choose a random initial policy π_0.
2: Set $\bar{\mu}_0 = \mu_0$ and $i = 1$.
3: **repeat**
4: Set $\alpha = \mu_E - \bar{\mu}_{i-1}$.
5: Compute an optimal policy π_i for the MDP with $R = \alpha^T \phi$
6: Compute an orthogonal projection of μ_E onto the line through $\bar{\mu}_{i-1}$ and μ_i.

$$\bar{\mu}_i = \bar{\mu}_{i-1} + \frac{(\mu_i - \bar{\mu}_{i-1})^T (\mu_E - \bar{\mu}_{i-1})}{(\mu_i - \bar{\mu}_{i-1})^T (\mu_i - \bar{\mu}_{i-1})} (\mu_i - \bar{\mu}_{i-1})$$

7: Set $t = \|\mu_E - \bar{\mu}_i\|_2$, and $i = i + 1$.
8: **until** $t \leq \varepsilon$

Output: the reward function R

function $R = \alpha^T \phi$ as follows :

$$V^{\pi}(s_0) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))|s_0\Big] = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \alpha^T \phi(s_t, \pi(s_t))|s_0\Big]$$
$$= \alpha^T \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \phi(s_t, \pi(s_t))|s_0\Big] = \alpha^T \mu(\pi),$$

where $\mu(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t, \pi(s_t))|s_0]$. Since the expert's policy is not explicitly given, the feature expectation of the expert's policy cannot be exactly computed. Thus, we empirically estimate the expert's feature expectation $\mu_E = \mu(\pi_E)$ from the given expert's M trajectories of the visited states $\{s_0^m, s_1^m, \cdots, s_{H-1}^m\}$ and the executed actions $\{a_0^m, a_1^m, \cdots, a_{H-1}^m\}$ by

$$\hat{\mu}_E = \frac{1}{M} \sum_{m=1}^{M} \sum_{t=0}^{H-1} \gamma^t \phi(s_t^m, a_t^m).$$

Abbeel and Ng (2004) propose apprenticeship learning algorithms for finding a policy whose value is similar to that of the expert's policy based on the idea that the difference of the values between the obtained policy π and the expert's policy π_E is bounded by the difference between their feature expectations. Formally, this is written as follows:

$$\begin{aligned} |V^{\pi_E}(s_0) - V^{\pi}(s_0)| &= |\alpha^T \mu(\pi_E) - \alpha^T \mu(\pi)| \\ &\leq \|\alpha\|_2 \|\mu_E - \mu(\pi)\|_2 \\ &\leq \|\mu_E - \mu(\pi)\|_2 \end{aligned} \tag{12}$$

since $\|\alpha\|_1$ is assumed to be bounded by 1. The algorithm is presented in Algorithm 2. The optimization problem in line 5 can be considered as the IRL step that tries to find the reward function that the expert is optimizing. It is similar to the optimization problem in Algorithm 1, except that, the optimization problem cannot be modeled as a linear programming (LP) problem but rather as a quadratically constrained programming (QCP) problem because of L_2 norm constraint on α. Algorithm 3 is an approximation algorithm using the projection method instead of QCP, where μ_i denotes $\mu(\pi_i)$ for all i. Both algorithms terminate when $t \leq \varepsilon$. It is proved that both algorithms take a finite number of iterations to terminate (Abbeel and Ng, 2004).

4. IRL in Partially Observable Environments

We denote the problem of IRL in partially observable environments as IRL for POMDP\R and the objective is to determine the reward function that the expert is optimizing. Formally, IRL for POMDP\R is defined as follows: Given a POMDP\R $\langle S, A, Z, T, O, b_0, \gamma \rangle$ and an expert's policy π_E, find the reward function R that makes π_E an optimal policy for the given POMDP. Hence, the reward function found by IRL for POMDP\R should guarantee the optimality of the expert's policy for the given POMDP.

IRL for POMDP\R mainly suffers from two sources: First, IRL is fundamentally ill-posed, and second, computational intractability arises in IRL for POMDP\R in contrast with IRL for MDP\R. We describe these challenges below.

An IRL problem is an ill-posed problem, which is a mathematical problem that is not well-posed. The three conditions of a well-posed problem are existence, uniqueness, and stability of the

solution (Hadamard, 1902). IRL violates the condition of the uniqueness. An IRL problem may have an infinite number of solutions since there may be an infinite number of reward functions that guarantee the optimality of the given expert's policy. A degenerate one is the solution of every IRL problem since $R = 0$ yields every policy optimal. Also, given an optimal policy for a reward function, we can find some other reward function that yields the same optimal policy without any modification to the environment by the technique of reward shaping (Ng et al., 1999).

As suggested by Ng and Russell (2000), we can guarantee the optimality of the expert's policy by comparing the value of the expert's policy with that of all possible policies. However, there are an infinite number of policies in a finite POMDP, since a policy in a POMDP is defined as a mapping from a continuous belief space to a finite state space or represented by an FSC policy that might have an infinite number of nodes. In contrast, there are a finite number of policies in a finite MDP, since a policy in an MDP is defined as a mapping from a finite state space to a finite action space. In addition, in order to compare two policies in a POMDP, the values of those policies should be compared for all beliefs, because the value function is defined on a belief space. This intractability of IRL for POMDP\R originates from the same cause as the difficulty of solving a POMDP. The optimal policy of a POMDP is the solution of a belief-state MDP using the concept of belief. It is then difficult to solve an MDP with a continuous state space, since a policy and its value function are respectively defined as a mapping from the continuous state space to the finite action space and the real numbers.

In the following sections, we address the problem of IRL for POMDP\R, considering two cases as in the approaches to IRL for MDP\R. The first case is when the expert's policy is explicitly represented in the form of an FSC. The second case is when the expert's policy is implicitly given by the trajectories of the expert's executed actions and the corresponding observations. Although the second case has more wide applicability than the first case, the first case can be applied to some practical problems. For example, when building dialogue management systems, we may already have a dialogue policy engineered by human experts, but we still do not know the reward function that produces the expert's policy. We propose several methods for the problems of IRL for POMDP\R in these two cases. For the first case, we formulate the problem with constraints for the reward functions that guarantee the optimality of the expert's policy. To address the intractability of IRL for POMDP\R, we derive conditions involving a small number of policies and exploiting the result of the classical POMDP research. For the second case, we propose iterative algorithms of IRL for POMDP\R. The motivation for this approach is from Ng and Russell (2000). We also extend the algorithms proposed by Abbeel and Ng (2004) to partially observable environments.

5. IRL for POMDP\R from FSC Policies

In this section, we present IRL algorithms for POMDP\R when the expert's policy is explicitly given. We assume that the expert's policy is represented in the form of an FSC, since the FSC is one of the most natural ways to specify a policy in POMDPs.

We propose three conditions for the reward function to guarantee the optimality of the expert's policy based on comparing Q-functions and using the generalized Howard's policy improvement theorem (Howard, 1960) and the witness theorem (Kaelbling et al., 1998). We then complete the optimization problems to determine a desired reward function.

5.1 Q-function Based Approach

We could derive a simple and naive condition for the optimality of the expert's policy by comparing the value of the expert's policy with those of all other policies. Given an expert's policy π_E defined by a directed graph $\langle \mathcal{N}, \mathcal{E} \rangle$,

$$V^{\pi_E}(n,b) \geq V^{\pi'}(n',b), \quad \forall b \in \Delta_n, \forall n' \in \mathcal{N}', \tag{13}$$

for all nodes $n \in \mathcal{N}$ and all other policies π' defined by a directed graph $\langle \mathcal{N}', \mathcal{E}' \rangle$, where Δ_n denotes the set of all the beliefs where node n is optimal. Since V^{π_E} and $V^{\pi'}$ are linear in terms of the reward function R by Equations (5) and (7), the above inequality yields the set of linear constraints that defines the feasible region of the reward functions that guarantees the expert's policy to be optimal.

However, enumerating all the constraints is clearly infeasible because we have to take into account all other policies π' including those with an infinite number of nodes, as well as all the infinitely many beliefs in Δ_n. In other words, Equation (13) yields infinitely many linear constraints. Hence, we propose a simple heuristic for choosing a finite subset of constraints that hopefully yields a tight specification of the feasible region for the true reward function. First, among the infinitely many policies, we only consider polices that are *slightly* modified from the expert's policy since they are *similar* to the expert's policy yet must be suboptimal. We select as the similar policies those deviate one step from the expert's action and observation strategies, analogous to Equation (8). For each node $n \in \mathcal{N}$, there are $|A||\mathcal{N}|^{|Z|}$ ways to deviate from the expert's action and observation strategies, hence we consider a total of $|\mathcal{N}||A||\mathcal{N}|^{|Z|}$ policies that deviate one step from the experts' policy. Second, instead of considering all possible beliefs in Δ_n, we only consider the finitely sampled beliefs reachable by the expert's policy. The motivation for using the sampled beliefs comes from the fact that only the set of beliefs reachable under the optimal policy is important for solving POMDPs, and it is also widely used in most of the recent approximate POMDP solvers (Spaan and Vlassis, 2005; Smith and Simmons, 2005; Pineau et al., 2006; Ji et al., 2007; Kurniawati et al., 2008).

The above heuristic yields the following finite set of linear constraints: Given an expert's policy $\pi_E = \langle \psi, \eta \rangle$,

$$Q^{\pi_E}(\langle n,b \rangle, \langle \psi(n), \eta(n,\cdot) \rangle) \geq Q^{\pi_E}(\langle n,b \rangle, \langle a, os \rangle), \quad \forall b \in B_n, \forall a \in A, \forall os \in \mathcal{N}^Z, \tag{14}$$

for every node n in π_E, where $B_n \subseteq \Delta_n$ denotes the set of sampled beliefs that are visited at node n when following the expert's policy π_E from the initial belief b_0. The above condition states that any policy that deviates one step from the expert's action and observation strategies should not have a higher value than the expert's policy does. Note that the condition is a necessary though not a sufficient one, since we do not use the set of all other policies but use the set of $|\mathcal{N}||A||\mathcal{N}|^{|Z|}$ policies that have the same (or possibly fewer) number of nodes as the expert's policy, nor do we use the set of all beliefs in Δ_n but use the set of sampled beliefs.

We use a simple example illustrating the approach. Consider a POMDP with two actions and two observations, and the expert's policy π_E is the FSC represented by solid lines in Figure 1. The nodes are labeled with actions (a_0 and a_1) and the edges are labeled with observations (z_0 and z_1). In order to find the region of the reward functions that yields π_E as optimal, we build one-step deviating policies as mentioned above. The policies $\pi'_0, \pi'_1, \cdots, \pi'_7$ in the figure are the one-step deviating policies for node n_0 of π_E. Note that π'_i visits node n'_i instead of the original node n_0 and then exactly follows π_E. We then enumerate the constraints in Equation (14), comparing the value of π_E to that

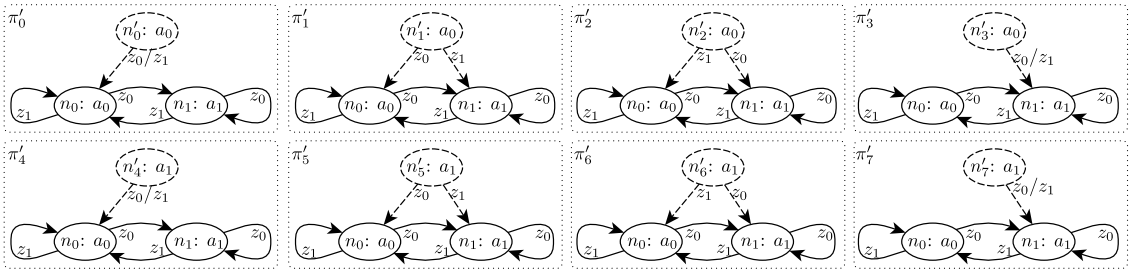

Figure 1: Set of the policies that deviate one step from node n_0 of the expert's policy.

of each one-step deviating policy. Specifically, the value at node n_0 of π_E is constrained to be not less than the value at node n'_i of π'_i, since deviating from the expert's policy should be suboptimal. To build the complete set of constraints in Equation (14), we additionally generate one-step deviating policies for node n_1 of π_E in a similar manner. We thus have $|\mathcal{N}||A||\mathcal{N}|^{|Z|} = 2 \times 2 \times 2^2 = 16$ policies that deviate one step from π_E.

5.2 Dynamic Programming (DP) Update Based Approach

A more systematic approach to defining the set of policies to be compared with the expert's policy is to use the set of FSC policies that arise during the DP update of the expert's policy. Given the expert's policy π_E, the DP update generates $|A||\mathcal{N}|^{|Z|}$ new nodes for all possible action and observation strategies, and these nodes can potentially be a new starting node. The expert's policy should be optimal if the value is not improved for any belief by the dynamic programming update. This idea comes from the generalized Howard's policy improvement theorem (Howard, 1960):

Theorem 2 [Hansen, 1998] *If an FSC policy is not optimal, the DP update transforms it into an FSC policy with a value function that is as good or better for every belief state and better for some belief state.*

The complete proof of the generalized policy improvement theorem can be found in Hansen (1998) but we give the full proof of the theorem for the convenience of the readers. First, we should prove the following lemma.

Lemma 1 *Given an FSC policy $\pi = \langle \psi, \eta \rangle$ and a node n_{new}, which is not included in π, the value function of node $n_{new} \in \mathcal{N}_{new}$ with the action strategy of selecting action a and observation strategy os is computed by*

$$V^{new}(n_{new}, s) = R(s,a) + \gamma \sum_{n',s'} T^{a,os}(\langle n_{new}, s \rangle, \langle n', s' \rangle) V^{\pi}(n', s'), \tag{15}$$

where V^{π} is calculated from Equation (5) and $T^{a,os}$ is defined in Equation (6). For some node n in π, if $V^{new}(n_{new}, s) \geq V^{\pi}(n, s)$, for $\forall s \in S$, the value of the original policy π will not be greater than that of the policy transformed by discarding node n and redirecting all the incoming edges of node n to node n_{new}.

Proof We build a new policy π_k that follows the original policy π, but executes the action and observation strategies of n_{new} for the first k times that node n is visited. The lemma is proved by the induction on the number of times k.

For the base step $k = 1$, the new policy π_1 executes the action and observation strategies of n_{new} only for the first time that node n is visited, and follows π for the rest of the time steps. Then, for any belief state b,

$$\begin{aligned} V^{\pi_1}(n,b) &= \sum_s b(s) V^{\pi_1}(n,s) = \sum_s b(s) V^{new}(n_{new},s) \\ &\geq \sum_s b(s) V^{\pi}(n,s) = V^{\pi}(n,b) \end{aligned}$$

since $V^{\pi_1}(n,s) = V^{new}(n_{new},s)$ for all $s \in S$ by the construction.

For the inductive step, we abuse notation to denote $R^{\pi_k}(s_t,a_t)$ as the reward at the t-th time step by following the policy π_k and starting from belief b and node n. Then, for any belief state b,

$$\begin{aligned} V^{\pi'_k}(n,b) &= \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R^{\pi_k}(s_t,a_t)\Big|b\Big] \\ &= \mathbb{E}\Big[\sum_{t=0}^{T_k-1} \gamma^t R^{\pi_k}(s_t,a_t) + \sum_{t=T_k}^{\infty} \gamma^t R^{\pi_k}(s_t,a_t)\Big|b\Big] \\ &= \mathbb{E}\Big[\sum_{t=0}^{T_k-1} \gamma^t R^{\pi_k}(s_t,a_t)\Big|b\Big] + \mathbb{E}\Big[\sum_{t=T_k}^{\infty} \gamma^t R^{\pi_k}(s_t,a_t)\Big|b\Big] \\ &= \mathbb{E}\Big[\sum_{t=0}^{T_k-1} \gamma^t R^{\pi_{k-1}}(s_t,a_t)\Big|b\Big] + \gamma^{T_k}\mathbb{E}\big[V^{\pi_k}(n,b_{T_k})|b\big] \\ &\geq \mathbb{E}\Big[\sum_{t=0}^{T_k-1} \gamma^t R^{\pi_{k-1}}(s_t,a_t)\Big|b\Big] + \gamma^{T_k}\mathbb{E}\big[V^{\pi}(n,b_{T_k})|b\big] \\ &= V^{\pi_{k-1}}(n,b) \end{aligned}$$

where T_k represents the k-th time that node n is visited. The first equality holds by the definition of the value function. The fourth equality holds by the construction of π_{k-1} and π_k and the definition of the value function. The fifth inequality holds by $V^{\pi_k}(n,b_{T_k}) = V^{new}(n_{new},b_{T_k}) \geq V^{\pi}(n,b_{T_k})$, since π_k executes the action and observation strategies of n_{new} at b_{T_k} and executes those of n for the rest of the time. Hence, by induction, it follows that the value of the transformed policy cannot be decreased by replacing n with n_{new}. ■

Using the above lemma, we can prove Theorem 2.

Proof (of Theorem 2) The policy iteration algorithm (Hansen, 1998) transforms the policy by replacing the nodes with new nodes generated by the DP update using the following rules: (1) If there is an old node whose action and observation strategies are the same as those of a new node, the old node is unchanged. (2) If the value at an old node is less than the value at a new node, for any state, the old node is discarded and all the incoming edges of the old node are redirected to the new node. (3) The rest of new nodes are added to the original policy.

Since the value is not decreased by leaving the policy unchanged or adding a node to the policy, the first and the third transformation rules cannot decrease the value. Also, by the above lemma, the second transformation rule cannot decrease the value. Thus, the value of the transformed policy using the DP update does not decrease.

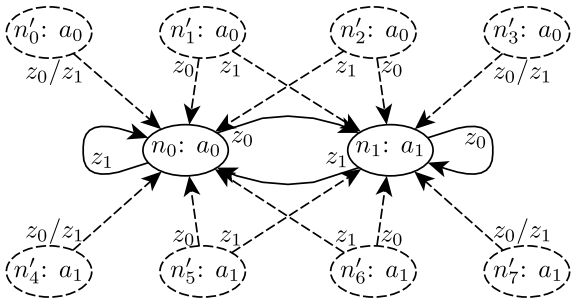

Figure 2: Set of the newly generated nodes by the DP update.

Also, if every node generated by the DP update is a duplicate of a node in the original policy, the optimality equation, Equation (4) is satisfied and the original policy is optimal. Thus, if the policy is not optimal, the DP update must generate some non-duplicate nodes that change the policy and improve the values for some belief state. ■

We should proceed with caution however in the sense that the DP update does not generate all the necessary nodes to guarantee the optimality of the expert's policy for every belief: The nodes in the expert's policy are only those reachable from the starting node n_0, which yields the maximum value at the initial belief b_0. Nodes that yield the maximum value at some other beliefs (i.e., useful) but are not reachable from n_0 are not present in the expert's policy. To guarantee the optimality of the expert's policy for every belief, we need to generate those non-existent but useful nodes. However, since there is no way to recover them, we only use nodes in the expert's policy and consider only the reachable beliefs by the expert's policy.

Let $\mathcal{N}_{new}$ be the set of nodes newly generated when transforming the expert's policy by the DP update, then $|\mathcal{N}_{new}| = |A||\mathcal{N}|^{|Z|}$. The value function of node $n_{new} \in \mathcal{N}_{new}$ is computed by Equation (15). The value function of policy π_E should satisfy

$$V^{\pi_E}(n,b) \geq V^{new}(n_{new},b), \quad \forall b \in B_n, \forall n_{new} \in \mathcal{N}_{new} \tag{16}$$

for every node $n \in \mathcal{N}$ if the expert's policy π_E is optimal. Note that V^{new} as well as V^{π_E} are linear in terms of the reward function R.

To illustrate the approach, we reuse the example in Section 5.1. Figure 2 shows π_E in solid lines and the set $\mathcal{N}_{new}$ of nodes generated by the DP update in dashed lines. We have $|A||\mathcal{N}|^{|Z|} = 2 \times 2^2 = 8$ nodes generated by the DP update, thus $\mathcal{N}_{new} = \{n'_0, n'_1, \cdots, n'_7\}$ is the complete set of nodes with all possible action and observation strategies. We then enumerate the constraints in Equation (16), making the value at each node of π_E no less than the values at the nodes in $\mathcal{N}_{new}$. Since the number of the newly generated nodes by the DP update is smaller than that of the policies generated by the Q-function based approach in Section 5.1, the computational complexity is significantly reduced.

5.3 Witness Theorem Based Approach

A more computationally efficient way to generate the set $\mathcal{N}_{new}$ of new nodes is to use the witness theorem (Kaelbling et al., 1998). We will exploit the witness theorem to find a set of useful nodes

that yield the feasible region for the true reward function as the witness algorithm incrementally generates new policy trees that improve the current policy trees. Here, we say that a node is useful if it has greater value than any other nodes at some beliefs. Formally speaking, given an FSC policy π, we define a set $B(n,U)$ of beliefs where the value function of node n dominates those of all other nodes in the set U:

$$B(n,U) = \{b | V^{new}(n,b) > V^{new}(n',b), \text{ for } \forall n' \in U \backslash \{n\}, \forall b \in \Delta\}$$

where $V^{new}(n,b) = \sum_s b(s) V^{new}(n,s)$ and $V^{new}(n,s)$ is computed by Equation (15). Node n is useful if $B(n,U) \neq \emptyset$, and U is a set of useful nodes if $B(n,U) \neq \emptyset$ for all $n \in U$. We re-state the witness theorem in terms of FSC policies as the following:

Theorem 3 [Kaelbling et al., 1998] *An FSC policy π is given as a directed graph $\langle \mathcal{N}, \mathcal{E} \rangle$. Let $\tilde{U}_a$ be a nonempty set of useful nodes with the action strategy of choosing action a, and U_a be the complete set of useful nodes with the action strategy of choosing action a. Then, $\tilde{U}_a \neq U_a$ if and only if there is some node $\tilde{n} \in \tilde{U}_a$, observation z^*, and node $n' \in \mathcal{N}$ for which there is a belief b such that*

$$V^{new}(n_{new}, b) \geq V^{\pi}(n,b)$$

for all $n \in \tilde{U}_a$, where n_{new} is a node that agrees with $\tilde{n}$ in its action and all its successor nodes except for observation z^*, for which $\eta(n_{new}, z^*) = n'$.

Proof The *if* direction of the statement is satisfied because b is a witness point for the existence of a useful node missing from $\tilde{U}_a$.

The *only if* direction can be rephrased as: If $\tilde{U}_a \neq U_a$ then there is a node $\tilde{n} \in \tilde{U}_a$, a belief state b, and a new node n_{new} that has a larger value than any other nodes $n \in \tilde{U}_a$ at b.

Choose some node $n^* \in U_a - \tilde{U}_a$. Since n^* is useful, there must be a belief b such that $V^{new}(n^*,b) > V^{new}(n',b)$ for all node $n' \in \tilde{U}_a$. Let $\tilde{n} = \text{argmax}_{n'' \in \tilde{U}_a} V^{new}(n'',b)$. Then, by the construction,

$$V^{new}(n^*,b) > V^{new}(\tilde{n},b). \tag{17}$$

Note that action a is always executed at n^* and $\tilde{n}$, since we consider only the nodes with the action strategy of choosing action a in the theorem.

Assume that for every observation z,

$$\begin{aligned} &\sum_s b(s) \sum_{s'} T(s,a,s') O(s',a,z) V^{\pi}(\eta(n^*,z), s') \\ &\qquad \leq \sum_s b(s) \sum_{s'} T(s,a,s') O(s',a,z) V^{\pi}(\eta(\tilde{n},z), s'). \end{aligned}$$

Then

$$\begin{aligned} V^{new}(n^*,b) &= \sum_s b(s) \left[R(s,a) + \gamma \sum_{s'} T(s,a,s') \sum_z O(s',a,z) V^{\pi}(\eta(n^*,z), s') \right] \\ &\leq \sum_s b(s) \left[R(s,a) + \gamma \sum_{s'} T(s,a,s') \sum_z O(s',a,z) V^{\pi}(\eta(\tilde{n},z), s') \right] \\ &= V^{new}(\tilde{n},b) \end{aligned}$$

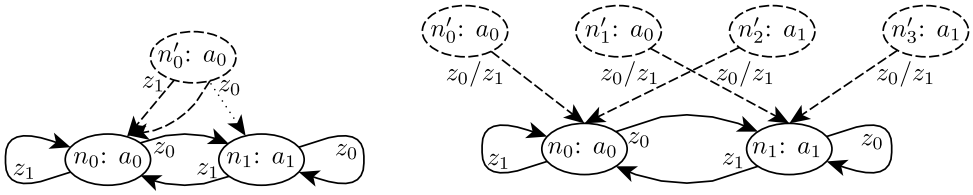

Figure 3: Set of the newly generated nodes by the witness theorem.

which contradicts (17). Thus, there must be some observation z^* such that

$$\sum_s b(s) \sum_{s'} T(s,a,s') O(s',a,z^*) V^\pi(\eta(n^*,z^*),s') > \sum_s b(s) \sum_{s'} T(s,a,s') O(s',a,z^*) V^\pi(\eta(\tilde{n},z^*),s').$$

Now, if $\tilde{n}$ and n^* differ in only one successor node, then the proof is complete with n^*, which can serve as the n_{new} in the theorem. If n and n^* differ in more than one successor node, we will identify another node that can act as n_{new}. Define n_{new} to be identical to $\tilde{n}$ except for observation z^*, for which $\eta(n_{new},z^*) = \eta(n^*,z^*)$. From this, it follows that

$$\begin{aligned} V^{new}(n_{new},b) &= \sum_s b(s) \left[R(s,a) + \gamma \sum_{s'} T(s,a,s') \sum_z O(s',a,z) V^\pi(\eta(n_{new},z),s') \right] \\ &> \sum_s b(s) \left[R(s,a) + \gamma \sum_{s'} T(s,a,s') \sum_z O(s',a,z) V^\pi(\eta(\tilde{n},z),s') \right] \\ &= V^{new}(\tilde{n},b) \geq V^{new}(n,b) \end{aligned}$$

for all $n \in \tilde{U}_a$. Therefore, the nodes $\tilde{n}$ and n_{new}, the observation z^*, $n' = \eta(n^*,z^*)$, and the belief state b satisfy the conditions of the theorem. ■

The witness theorem tells us that if a policy π is optimal, then the value of n_{new} generated by changing the successor node of each single observation should not increase for any possible beliefs. This leads us to a smaller set of inequality constraints compared to Equation (16), by defining $\mathcal{N}_{new}$ in a different way.

Let $\mathcal{N}_a = \{n \in \mathcal{N} | \psi(n) = a\}$ and $A_{-\mathcal{N}} = \{a \in A | \mathcal{N}_a = \emptyset\}$. For each action $a \notin A_{-\mathcal{N}}$, we generate new nodes by the witness theorem: For each node $\tilde{n} \in \mathcal{N}_a$, $z^* \in Z$, and $n' \in \mathcal{N}$, we make n_{new} such that $\psi(n_{new}) = \psi(\tilde{n}) = a$ and $\eta(n_{new},z) = \eta(\tilde{n},z)$ for all $z \in Z$ except for z^*, for which $\eta(n_{new},z^*) = n'$. The maximum number of newly generated nodes by the witness theorem is $\sum_a |\mathcal{N}_a||\mathcal{N}||Z| \leq |\mathcal{N}|^2|Z|$. Then, for each action $a \in A_{-\mathcal{N}}$, we use the DP update to generate $|A_{-\mathcal{N}}||\mathcal{N}|^{|Z|}$ additional nodes. The number of newly generated nodes $|\mathcal{N}_{new}|$ is no more than $|\mathcal{N}|^2|Z| + |A_{-\mathcal{N}}||\mathcal{N}|^{|Z|}$. Note that this number is often much less than $|A||\mathcal{N}|^{|Z|}$, the number of newly generated nodes by DP update, since the number of actions $|A_{-\mathcal{N}}|$ that is not executed at all by the expert's policy is typically much fewer than $|A|$.

We again reuse the example in Section 5.1 to illustrate the approach. We build the set $\mathcal{N}_{new}$ of new nodes using the witness theorem. The left panel of Figure 3 shows the construction of

$$\begin{aligned}
\text{maximize}_R \quad & \sum_{n\in\mathcal{N}}\sum_{b\in B_n}\sum_{\substack{a\in A\setminus\psi(n)\\ os\in\mathcal{N}^Z\setminus\eta(n,\cdot)}}\Big[V^\pi(\langle n,b\rangle)-Q^\pi(\langle n,b\rangle,\langle a,os\rangle)\Big]-\lambda\|R\|_1\\
\text{subject to} \quad & Q^\pi(\langle n,b\rangle,\langle\psi(n),\eta(n,\cdot)\rangle)\geq Q^\pi(\langle n,b\rangle,\langle a,os\rangle),\\
& \qquad\qquad \forall b\in B_n,\forall a\in A,\forall os\in\mathcal{N}^Z,\forall n\in\mathcal{N}\\
& |R(s,a)|\leq R_{max},\quad \forall s,\forall a
\end{aligned}$$

Table 2: Optimization problem using Q-function based optimality constraint.

$$\begin{aligned}
\text{maximize}_R \quad & \sum_{n\in\mathcal{N}}\sum_{b\in B_n}\sum_{n_{new}\in\mathcal{N}_{new}}\Big[V^\pi(n,b)-V^{new}(n_{new},b)\Big]-\lambda\|R\|_1\\
\text{subject to} \quad & V^\pi(n,b)\geq V^{new}(n_{new},b),\quad \forall b\in B_n,\forall n_{new}\in\mathcal{N}_{new},\forall n\in\mathcal{N}\\
& |R(s,a)|\leq R_{max},\quad \forall s,\forall a
\end{aligned}$$

Table 3: Optimization problem using the DP update or the witness theorem based optimality constraint.

new node n_0' from node n_0 such that $\psi(n_0')=\psi(n_0)=a_0$ and $\eta(n_0',z_1)=\eta(n_0,z_1)$. The original observation strategy of n_0 for z_0 transits to n_1 (shown in dotted line), and it is changed to n_0 (shown in dashed line). The right panel in the figure presents the complete set $\mathcal{N}_{new}$ of generated nodes using the witness theorem (shown in dashed lines). Nodes n_0' and n_1' are generated from node n_0, whereas nodes n_2' and n_3' are from node n_1. Note that $A_{-\mathcal{N}}=\emptyset$ since π_E executes all actions in the model. We thus have a total of 4 generated nodes, which is smaller than those generated by either the Q-function based or the DP update based approach.

5.4 Optimization Problem

In the previous sections, we suggested three constraints for the reward function that stem from the optimality of the expert's policy, but infinitely many reward functions can satisfy the constraints in Equations (14) and (16). We thus present constrained optimization problems with objective functions that encode our preference on the learned reward function. As in Ng and Russell (2000), we prefer a reward function that maximizes the sum of the margins between the expert's policy and other policies. At the same time, we want the reward function as sparse as possible, which can be accomplished by adjusting the penalty weight on the L_1-norm of the reward function. If we use Q-function based optimality constraint, that is, Equation (14), the value of the expert's policy is compared with those of all other policies that deviate from the expert's action and observation strategies, given in Table 2. When using the DP update or the witness theorem based optimality constraint, that is, Equation (16), the policies other than the expert's policy are captured in newly generated nodes n_{new}, hence the optimization problem now becomes the one given in Table 3. Since all the inequalities and the objective functions in the optimization problems are linear in terms

of the reward function, the desired reward function can be found efficiently by solving the linear programming problems.

When using Q-function or the DP update based approach, the number of policies compared with the expert's is exponential to the number of observations, and hence the number of constraints in the optimization problems increases exponentially. This may become intractable even for a small size expert's policy. We can address this limitation using the witness theorem based approach, since it is sufficient to consider as few as $|\mathcal{N}|^2|Z|$ nodes if the expert's policy executes all actions, which is common in many POMDP benchmark problems.

6. IRL for POMDP\R from Sampled Trajectories

In some cases, the expert's policy may not be explicitly given, but the records of the expert's trajectories may be available instead.[4] Here, we assume that the set of H-step belief trajectories is given. The m-th trajectory is denoted by $\{b_0^m, b_1^m, \ldots, b_{H-1}^m\}$, where $b_0^m = b_0$ for all $m \in \{1, 2, \cdots, M\}$. If the trajectories of the perceived observations $\{z_0^m, z_1^m, \ldots, z_{H-1}^m\}$ and the executed actions $\{a_0^m, a_1^m, \ldots, a_{H-1}^m\}$ following the expert's policy are available instead, we can reconstruct the belief trajectories by using the belief update in Equation (3).

In order to obtain an IRL algorithm for POMDP\R from the sampled belief trajectories, we linearly parameterize the reward function using the known basis functions $\phi : S \times A \to [0,1]^d$ and the weight vector $\alpha \in [-1,1]^d$ as in Equation (11): $R(s,a) = \alpha^T \phi(s,a)$. This assumption is useful for the problems with large state spaces, because with some prior knowledge about the problems, we can represent the reward function compactly using the basis functions. For example, in robot navigation problems, the basis function can be chosen to capture the features of the state space, such as which locations are considered dangerous. In the worst case when no such prior knowledge is available, the basis functions may be designed for each pair of state and action so that the number of basis functions is $|S| \times |A|$. The objective of IRL is then to determine the (unknown) parameter α of the reward function $R = \alpha^T \phi$.

In this section, we propose three trajectory-based IRL algorithms for POMDP\R. The algorithms share the same framework that iteratively repeats estimating the parameter of the reward function using an IRL algorithm and computing an optimal policy for the estimated reward function using a POMDP solver. The first algorithm finds the reward function that maximizes the margin between the values of the expert's policy and other policies for the sampled beliefs using LP. This is a simple extension to Ng and Russell (2000). The second algorithm computes the reward function that maximizes the margin between the feature expectations of the expert's policy and other policies using QCP. The last algorithm approximates the second using the projection method. The second and third algorithms are extended from the methods originally suggested for MDP environments by Abbeel and Ng (2004).

4. As in the IRL for MDP\R from sampled trajectories, we assume that the transition and observation functions are known in POMDP\R.

Algorithm 4 IRL for POMDP\R from the sampled trajectories using the MMV method.

Input: POMDP\R $\langle S, A, Z, T, O, b_0, \gamma \rangle$, basis functions ϕ, M trajectories

Choose a set B^{π_E} of all the unique beliefs in the trajectories.
Choose a random initial policy π_1 and set $\Pi = \{\pi_1\}$.
for $k = 1$ to $MaxIter$ **do**
    Find $\hat{\alpha}$ by solving the linear program:

$$\begin{aligned} &\text{maximize}_{\hat{\alpha}} \quad \sum_{\pi \in \Pi} \sum_{b \in B^{\pi_E}} p\left(\hat{V}^{\pi_E}(b) - V^{\pi}(b)\right) - \lambda \|\hat{\alpha}^T \phi\|_1 \\ &\text{subject to} \quad |\hat{\alpha}_i| \leq 1, \quad i = 1, 2, \cdots, d \end{aligned}$$

    Compute an optimal policy π_{k+1} for the POMDP with $\hat{R} = \hat{\alpha}_k^T \phi$.
    if $|\hat{V}^{\pi_E}(b) - V^{\pi_{k+1}}(b)| \leq \varepsilon$, $\forall b \in B^{\pi_E}$ **then**
        return $\hat{R} = \hat{\alpha}_k^T \phi$
    else
        $\Pi = \Pi \cup \{\pi_{k+1}\}$
    end if
end for
$K = \text{argmin}_{k:\pi_k \in \Pi} \max_{b \in B^{\pi_E}} |\hat{V}^{\pi_E}(b) - V^{\pi_k}(b)|$
return $\hat{R} = \hat{\alpha}_K^T \phi$

Output: the reward function $\hat{R}$

6.1 Max-Margin between Values (MMV) Method

We first evaluate the values of the expert's policy and other policies for the weight vector α of a reward function in order to compare their values. The reward for belief b is then calculated by

$$R(b,a) = \sum_{s \in S} b(s) R(s,a) = \sum_{s \in S} b(s) \alpha^T \phi(s,a) = \alpha^T \phi(b,a),$$

where $\phi(b,a) = \sum_{s \in S} b(s) \phi(s,a)$. We also compute $\hat{V}^{\pi_E}(b_0^m)$ to be the empirical return of the expert's m-th trajectory by

$$\hat{V}^{\pi_E}(b_0^m) = \sum_{t=0}^{H-1} \gamma^t R(b_t^m, a_t^m) = \sum_{t=0}^{H-1} \gamma^t \alpha^T \phi(b_t^m, a_t^m).$$

Noting that $b_0^m = b_0$ for all m, the expert's average empirical return at b_0 is given by

$$\hat{V}^{\pi_E}(b_0) = \frac{1}{M} \sum_{m=1}^{M} \hat{V}^{\pi_E}(b_0^m) = \alpha^T \frac{1}{M} \sum_{m=1}^{M} \sum_{t=0}^{H-1} \gamma^t \phi(b_t^m, a_t^m), \tag{18}$$

which is linear in terms of α. In a similar manner, we can compute the average empirical return of the expert's trajectories at other beliefs b_j by

$$\hat{V}^{\pi_E}(b_j) = \frac{1}{M_j} \sum_{m=1}^{M} \hat{V}^{\pi_E}(b_j) = \alpha^T \frac{1}{M_j} \sum_{m=1}^{M} \sum_{t=H_j^m}^{H-1} \gamma^{t-H_j^m} \phi(b_t^m, a_t^m), \tag{19}$$

where H_j^m is the first time that b_j is found in the m-th trajectory and M_j is the number of trajectories that contain b_j.

Given the above definitions, the rest of the derivation is fairly straightforward, and leads to a similar algorithm to that of Ng and Russell (2000). The algorithm is shown in Algorithm 4. It iteratively tries to find a reward function parameterized by α that maximizes the sum of the margins between the value $\hat{V}^{\pi_E}$ of the expert's policy and the value V^{π} of each FSC policy $\pi \in \Pi$ found so far by the algorithm at all the unique beliefs $b \in B^{\pi_E}$ in the trajectories. We could consider the initial belief b_0 alone, similar to Ng and Russell (2000) considering the initial state s_0 alone. However, we found it more effective in our experiments to include additional beliefs, since they often provide better guidance in the search of the reward function by tightening the feasible region. In order to consider the additional beliefs, we should be able to compute the value V^{π} of the intermediate policy π at belief $b \in B^{\pi_E}$, but it is not well defined. b may be unreachable under π and it is not known that we will visit b at which node of π. In our work, we use an upperbound approximation given as

$$V^{\pi}(b) \approx \max_{n} V^{\pi}(n, b), \tag{20}$$

where $V^{\pi}(n,b)$ is computed by Equation (7).

The IRL step in line 4 finds the reward function that guarantees the optimality of the expert's policy. In the optimization problem, we constrain the value of the expert's policy to be greater than that of other policies in order to ensure that the expert's policy is optimal, and maximize the sum of the margins between the expert's policy and other policies using a monotonically increasing function p.[5] In addition, we prefer the sparse reward function and the sparsity of the learned reward function can be achieved by tuning the penalty weight λ. Note that we can solve the IRL step in Algorithm 4 using LP since all the variables such as $\hat{V}^{\pi_E}$ and V^{π} are linear functions in terms of α from Equations (18), (19), and (20).

When π_{k+1} matches π_E, the differences in the value functions for all beliefs will vanish. Hence, the algorithm terminates when all the differences in the values are below the threshold ε, or the iteration number has reached the maximum number of steps *MaxIter* to terminate the algorithm in a finite number of iterations.

6.2 Max-Margin Between Feature Expectations (MMFE) Method

We can re-write the value of a FSC policy π in POMDPs using the feature expectation $\mu(\pi)$, proposed by Abbeel and Ng (2004) as follows:

$$\begin{aligned} V^{\pi}(b_0) &= \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(b_t, a_t) | \pi, b_0\Big] = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \alpha^T \phi(b_t, a_t) | \pi, b_0\Big] \\ &= \alpha^T \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \phi(b_t, a_t) | \pi, b_0\Big] = \alpha^T \mu(\pi), \end{aligned}$$

where $\mu(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(b_t, a_t) | \pi, b_0]$, and it is assumed that $\|\alpha\|_1 \leq 1$ to bound R_{max} by 1. In order to compute the feature expectation $\mu(\pi)$ exactly, we define the occupancy distribution $occ^{\pi}(s,n)$ of the policy π that represents the relative frequency of visiting state s at node n when following the policy $\pi = \langle \psi, \eta \rangle$ and starting from belief b_0 and node n_0. It can be calculated by solving the

5. We simply choose $p(x) = x$ if $x > 0$ and $p(x) = 2x$ if $x < 0$ as in Ng and Russell (2000). This gives more penalty to violating the optimality of the expert's policy.

Algorithm 5 IRL for POMDP\R from the sampled trajectories using the MMFE method.

Input: POMDP\R $\langle S, A, Z, T, O, b_0, \gamma \rangle$, basis functions ϕ, M trajectories

Choose a random initial weight α_0.
$\Pi = \emptyset$, $\Omega = \emptyset$, and $t = \infty$.
for $k = 1$ to $MaxIter$ **do**
  Compute an optimal policy π_{k-1} for the POMDP with $R = \alpha_{k-1}^T \phi$.
  $\Pi = \Pi \cup \{\pi_{k-1}\}$ and $\Omega = \Omega \cup \{\alpha_{k-1}\}$.
  if $t \leq \varepsilon$ **then**
    break
  end if
  Solve the following optimization problem:

$$\begin{aligned} &\text{maximize}_{t,\alpha_k} \quad t \\ &\text{subject to} \quad \alpha_k^T \mu_E \geq \alpha_k^T \mu(\pi) + t, \quad \forall \pi \in \Pi \\ &\qquad\qquad \|\alpha_k\|_2 \leq 1 \end{aligned}$$

end for
$K = \text{argmin}_{k:\pi_k \in \Pi} \|\mu_E - \mu(\pi_k)\|_2$
return $R = \alpha_K^T \phi$

Output: the reward function R

following system of linear equations:

$$\begin{aligned} occ^{\pi}(s', n') = b_0(s', n')\delta_{n', n_0} + \\ \gamma \sum_{s,z,n} occ^{\pi}(s, n) T(s, \psi(n), s') O(s', \psi(n), z) \delta_{n', \eta(n,z)}, \quad \forall s' \in S, \forall n' \in \mathcal{N}, \end{aligned}$$

where $\delta_{x,y}$ denotes the Kronecker delta function, defined as $\delta_{x,y} = 1$ if $x = y$ and $\delta_{x,y} = 0$ otherwise. With the occupancy distribution, the value of the policy π can be computed by

$$V^{\pi}(b_0) = \sum_{s,n} occ^{\pi}(s, n) R(s, \psi(n)) = \sum_{s,n} occ^{\pi}(s, n) \alpha^T \phi(s, \psi(n)) = \alpha^T \mu(\pi),$$

where $\mu(\pi) = \sum_{s,n} occ^{\pi}(s, n) \phi(s, \psi(n))$. However, the feature expectation of the expert's policy π_E cannot be exactly computed, because we only have the set of trajectories on the belief space, which are recovered from the given trajectories of the actions and the observations, instead of the explicit FSC form of the expert's policy. Hence, we estimate the expert's feature expectation $\mu(\pi_E) = \mu_E$ empirically by

$$\hat{\mu}_E = \frac{1}{M} \sum_{m=1}^{M} \sum_{t=0}^{H-1} \gamma^t \phi(b_t^m, a_t^m).$$

From these definitions, we can derive the following inequalities, which are similar to Equation (12),

$$\begin{aligned} |V^{\pi_E}(b_0) - V^{\pi}(b_0)| &= |\alpha^T \mu_E - \alpha^T \mu(\pi)| \\ &\leq \|\alpha\|_2 \|\mu_E - \mu(\pi)\|_2 \\ &\leq \|\mu_E - \mu(\pi)\|_2. \end{aligned} \tag{21}$$

Algorithm 6 IRL for POMDP\R from the sampled trajectories using the PRJ method.

Input: POMDP\R $\langle S, A, Z, T, O, b_0, \gamma \rangle$, basis functions ϕ, M trajectories

1: Choose a random initial weight α_0.
2: Compute an optimal policy π_0 for the POMDP $R = \alpha_0^T \phi$.
3: $\Pi = \{\pi_0\}$, $\Omega = \{\alpha_0\}$, $\bar{\mu}_0 = \mu_0$ and $t = \infty$.
4: **for** $k = 1$ to *MaxIter* **do**
5: $\quad \alpha_k = \mu_E - \bar{\mu}_{k-1}$.
6: $\quad$ Compute an optimal policy π_k for the POMDP $R = \alpha_k^T \phi$.
7: $\quad \Pi = \Pi \cup \{\pi_k\}$ and $\Omega = \Omega \cup \{\alpha_k\}$.
8: $\quad$ **if** $t \leq \varepsilon$ **then**
9: $\quad\quad$ break
10: $\quad$ **end if**
11: $\quad$ Compute an orthogonal projection of μ_E onto the line through $\bar{\mu}_{k-1}$ and μ_k

$$\bar{\mu}_k = \bar{\mu}_{k-1} + \frac{(\mu_k - \bar{\mu}_{k-1})^T (\mu_E - \bar{\mu}_{k-1})}{(\mu_k - \bar{\mu}_{k-1})^T (\mu_k - \bar{\mu}_{k-1})} (\mu_k - \bar{\mu}_{k-1})$$

12: $\quad t = \|\mu_E - \bar{\mu}_k\|_2$
13: **end for**
14: $K = \text{argmin}_{k:\pi_k \in \Pi} \|\mu_E - \mu_k\|_2$.
15: **return** $R = \alpha_K^T \phi$

Output: the reward function R

The last inequality holds since we assume $\|\alpha\|_1 \leq 1$. The above inequalities state that the difference between the expert's policy π_E and any policy π is bounded by the difference between their feature expectations, which is the same result as in Abbeel and Ng (2004). Based on Equation (21), we can easily extend Algorithm 2 to address the IRL problem for POMDP\R from the sampled trajectories. The algorithm is presented in Algorithm 5. While we can solve Algorithm 4 using LP, the algorithm requires a QCP solver, since the optimization problem in line 9 has a 2-norm constraint on α. Note that it is proved that the algorithm will terminate in a finite number of iterations in Abbeel and Ng (2004).

Abbeel and Ng (2004) construct a policy by mixing the policies found by the algorithm in order to find the policy that is as good as the given expert's policy. They choose the weight of the policies by computing the convex combination of feature expectations that minimizes the distance to the expert's feature expectation. However, this method cannot be adapted to our IRL algorithm, because there is no way to recover the reward function that provides the computed mixed policy. Thus, we return the reward function that yields the closest feature expectation to that of the expert's policy among the intermediate reward functions found by the algorithm. By Equation (21), the value of the policy that generates the closest feature expectation is assured to be similar to the value of the expert's policy and we hope that the reward function that yields the closest feature expectation will be similar to the reward function that the expert is optimizing.

6.3 Projection (PRJ) Method

In the previous section, we described the IRL algorithm for POMDP\R from the sampled trajectories using QCP. We can now address the problem using a simpler method, as Abbeel and Ng (2004) proposed. The IRL step in Algorithm 5 can be considered for finding the unit vector μ_k or-

thogonal to the maximum margin hyperplane that classifies feature expectations into two sets: One set consists of the expert's feature expectation and the other set consists of the feature expectations of the policies found by the algorithm. The unit vector μ_k can then be approximately computed by projecting the expert's feature expectation on the line between the feature expectations of the most recent policy and the previously projected point. The algorithm is shown in Algorithm 6. In the algorithm, μ_i denotes $\mu(\pi_i)$ for all i and $\bar{\mu}$ denotes the point where the expert's feature expectation is projected. Similar to Algorithm 5, the algorithm returns the reward function that yields the closest feature expectation to that of the expert's policy among the intermediate reward functions found by the algorithm.

7. Experimental Results

In this section, we present the results from the experiments on some POMDP benchmark domains - *Tiger*, *1d Maze*, *5×5 Grid World*, *Heaven/Hell*, and *Rock Sample* problems. The characteristics of each problem is presented in Table 4 and brief explanations are given below.

The *Tiger* and *1d Maze* problems are classic POMDP benchmark problems (Cassandra et al., 1994). In the *Tiger* problem, an agent is standing in front of two doors. There is a reward behind one of the doors and a tiger behind the other. If the agent opens the door with the tiger, it gets a large penalty (-100). Otherwise, it receives the reward (+10). The agent initially does not know the location of the tiger. It can infer the location of the tiger by listening for the sound of the tiger with a small cost (-1) and the correct information is given with some probability (0.85). In the *1d Maze* problem, there are 4 states as presented in the first panel of Figure 4. The third state from the left is the goal state. An agent is initially set to the non-goal states with equal probabilities and can move left or right. The agent observes whether it is at the goal state or not. When the agent reaches the goal state, it is randomly moved to a non-goal state after executing any action.

The 5×5 *Grid World* problem is inspired by a problem in Ng and Russell (2000), where the states are located as shown in the second panel of Figure 4. An agent can move west, east, north or south, and their effects are assumed to be deterministic. The agent always starts from the north-west corner of the grid and the goal is at the south-east corner. After the agent reaches the goal state, the agent restarts from the start state by executing any action in the goal state. The current position cannot be observed directly but the presence of the adjacent walls can be perceived without noise. Hence, there are nine observations, eight of them corresponding to eight possible configurations of the nearby walls when on the border (N, S, W, E, NW, NE, SW, and SE), and one corresponding to no wall observation when not on the border (Null).

The *Heaven/Hell* problem (Geffner and Bonet, 1998) is a navigation problem over the states depicted in the third panel of Figure 4. The goal state is either position 4 or 6. One of these is heaven and the other is hell. When the agent reaches heaven, it receives a reward (+1). When it reaches hell, it receives a penalty (-1). It starts at position 0, and does not know the position of heaven. However, it can get the information about the position of heaven after visiting the priest at position 9. The agent always perceives its current position without any noise. After reaching heaven or hell, it is moved at the initial position.

The *Rock Sample* problem (Smith and Simmons, 2004) models a rover that moves around an area and samples rocks. The locations of the rover and the rocks are known (the rocks are marked with stars in the fourth panel of Figure 4), but the value of the rocks are unknown. If it samples a good rock, it receives a reward (+10), but if it samples a bad rock, it receives a penalty (-10). When

Problem	$\|S\|$	$\|A\|$	$\|Z\|$	γ	$\|\mathcal{N}\|$	$\|\cup_{n\in\mathcal{N}} B_n\|$
Tiger	2	3	2	0.75	5	5
1d Maze	4	2	2	0.75	3	4
5×5 *Grid World*	25	4	9	0.90	2	13
Heaven/Hell	20	4	11	0.99	18	19
Rock Sample[4,3]	129	8	2	0.95	16	22

Table 4: Characteristics of the problem domains used in the experiments. γ: The discount factor. $|\mathcal{N}|$: The number of nodes in the optimal policy. $|\cup_{n\in\mathcal{N}} B_n|$: The total number of beliefs reachable by the optimal policy.

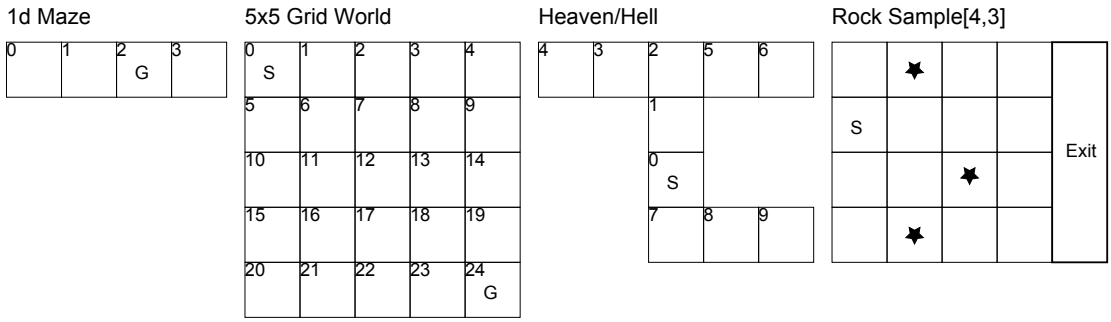

Figure 4: Maps for the *1d Maze*, 5×5 *Grid World*, *Heaven/Hell*, and *Rock Sample*[4,3] problems.

the rover tries to sample at the location without any rocks, it receives a large penalty (-100). The rover can observe the value of the rocks with a noisy long range sensor. In addition, it gets a reward (+1) if it reaches the right side of the map. When it reaches other sides of the map, it gets a large penalty (-100). The rover is immediately moved to the start position when it traverses outside of the map. The *Rock Sample* problem is instantiated as *Rock Sample*$[n,k]$, which describes that the size of the map is $n\times n$ and the number of the rocks on the map is k, and our experiment was performed on *Rock Sample*[4,3].

To evaluate the performance of the IRL algorithms, we could naively compare the true reward functions in the original problems to the reward functions found by the algorithms. However, it is not only difficult but also meaningless to simply compare the numerical values of the reward functions, since the reward function represents the relative importance of executing an action in a state. Completely different behaviors may be derived from two reward functions that have a small difference, and an identical optimal policy may be induced by two reward functions that have a large difference. For example, three reward functions in the *Tiger* problem are presented in Table 5, where R^* is the true reward function and R_1 and R_2 are two reward functions chosen for explaining the phenomenon. When the distances are measured by L_2 norm,

$$Dist(R,R^*) = \|R^*-R\|_2 = \sqrt{\sum_{s\in S,a\in A}(R^*(s,a)-R(s,a))^2},$$

the reward function R_2 is more similar to R^* than the reward function R_1. However, as shown in Figure 5, the optimal policies for R^* and R_1 are exactly the same while the optimal policy for R_2

	Listen	*Success*	*Failure*	$Dist(R,R^*)$	$V^\pi(b_0;R^*)$
R^*	-1	10	-100	0	1.93
R_1	-1	5.68	-100	6.10	1.93
R_2	3.05	10	-100	5.73	1.02

Table 5: Three reward functions in the *Tiger* problem. R^* is the true reward function. *Listen*: The negative cost of listening. *Success*: The reward of opening the correct door. *Failure*: The negative penalty of choosing the door with the tiger. $Dist(R,R^*)$: The distance from the true reward functions. $V^\pi(b_0;R^*)$: The value of the optimal policy for each reward function measured on the true reward function.

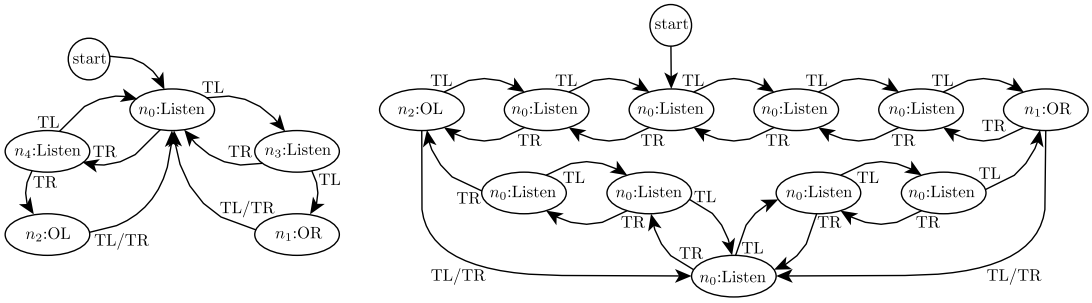

Figure 5: Optimal policies for the reward functions in Table 5. The nodes are labeled with actions (Listen, OL: Open-left, OR: Open-right). The edges are labeled with observations (TL: Tiger-left, TR: Tiger-right). *Left*: The optimal policy for R^* and R_1. *Right*: The optimal policy for R_2.

is different from that for R^*. If we still want to directly evaluate the computed reward function using a distance measure, we could apply the policy-invariant reward transformation on the true reward function and compute the minimum distance, but it is non-trivial to do so since there is an infinite number of transformations to choose from including the positive linear transformation and the potential-based shaping (Ng et al., 1999). Therefore, we compare the value functions of the optimal policies induced from the true and learned reward functions instead of directly measuring the distance between the reward functions.

The performance of the algorithms are evaluated by the differences in the values of the expert's policy and the optimal policy for the learned reward function. In the evaluations, the value of each policy is measured on the true reward function R^* and the learned reward function R_L, and we define the value $V^\pi(b_0;R)$ of a policy π at the initial belief b_0 measured on a reward function R as

$$V^\pi(b_0;R) = \sum_{s\in S} b_0(s) V^\pi(n_0,s;R),$$

where n_0 is the starting node of a policy π and $V^\pi(n_0,s;R)$ is computed by Equation (5) using the reward function R.

Problem	$D(R^*)$	$D(R_L)$	$\lvert\mathcal{N}_{new}\rvert$			Time		
			Q-IRL	D-IRL	W-IRL	Q-IRL	D-IRL	W-IRL
Tiger	0	0	375	75	39	0.07	0.04	0.03
1d Maze	0	0	54	18	12	0.02	0.02	0.02
5×5 *Grid World*	0	0	4096	2048	1044	78.06	3.00	1.54
Heaven/Hell	0	0	4.63×10^{15}	2.57×10^{14}	3260	n.a.	n.a.	6.20
Rock Sample[4,3]	13.42	0	32768	2048	634	77.99	19.20	3.77

n.a. = not applicable

Table 6: Results of IRL for POMDP\R from FSC policies. Q-IRL, D-IRL, and W-IRL respectively denote the Q-function based approach, the DP update based approach, and the witness theorem based approach. $D(R^*) = |V^{\pi_E}(b_0;R^*) - V^{\pi_L}(b_0;R^*)|$. $D(R_L) = |V^{\pi_E}(b_0;R_L) - V^{\pi_L}(b_0;R_L)|$. $|\mathcal{N}_{new}|$ denotes the number of newly generated policies. The average computation times are reported in seconds

Our algorithm requires a POMDP solver for computing the expert's policy and the intermediate optimal policies of the learned rewards. Since we assume the policy is in the form of an FSC, we use PBPI (Ji et al., 2007), which finds an optimal FSC policy approximately on the reachable beliefs. Optimization problems formulated in LP and QCP are solved using ILOG CPLEX.

The experiments are organized into two cases according to the representation of the expert's policy. In the first case, the expert's policy is explicitly given in the form of a FSC, and in the second case, the trajectories of the expert's executed actions and the corresponding observations are given instead.

7.1 Experiments on IRL from FSC Policies

The first set of experiments concerns the case in which the expert's policy is explicitly given using the FSC representation. We experimented with all three approaches in Section 5: The Q-function based approach, the DP update based approach, and the witness theorem based approach. As in the case of IRL for MDP\R, we were able to control the sparseness in the reward function by tuning the penalty weight λ. With a suitable value for λ, all three approaches yielded the same reward function.[6]

A summary of the experiments is given in Table 6. Since the *Heaven/Hell* problem has a larger number of observations than other problems and the Q-function and the DP update based approaches generate exponentially many new policies with respect to the number of observations, the optimization problems of the Q-function and the DP update based approaches were not able to handle the *Heaven/Hell* problem. Hence, the *Heaven/Hell* problem could only be solved by the witness theorem based approach. Also, the witness theorem based approach was able to solve the other problems more efficiently than the Q-function based approach and the DP update based approach.

6. With any value of λ, the reward functions computed by all the proposed optimization problems should guarantee the optimality of the expert's policy, except for the degenerated case $R = 0$ due to an overly large value of λ. However, we observed that the optimality of our solutions is often subject to numerical errors in the optimization, which is an interesting issue for future studies.

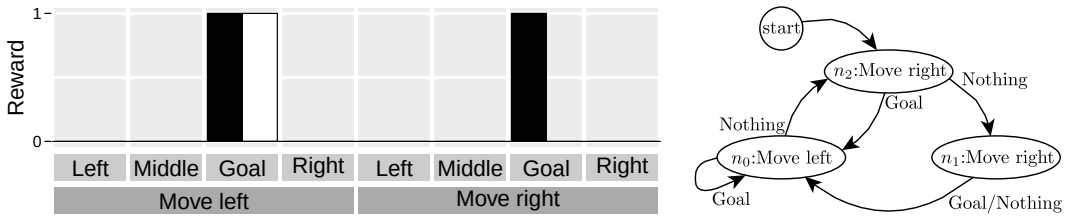

Figure 6: Comparison of the true and learned reward functions and the expert's policy in the *1d Maze* problem. *Black bars*: The true reward function. *White bars*: The learned reward function.

In Table 6, $D(R^*) = |V^{\pi_E}(b_0;R^*) - V^{\pi_L}(b_0;R^*)|$ is the difference between the values of the expert's policy π_E and the optimal policy π_L for the learned reward, which are measured on the true reward function R^*. $D(R_L) = |V^{\pi_E}(b_0;R_L) - V^{\pi_L}(b_0;R_L)|$ is the difference between the values measured on the learned reward function R_L. The differences measured on the true reward function in the *Tiger*, *1d Maze*, 5×5 *Grid World*, and *Heaven/Hell* are zero, meaning that the learned reward function generated a policy whose performance is the same as that of the expert's policy. However, our algorithms failed to find the reward that generates a policy that is optimal on the true reward in the *Rock Sample*[4,3]. Nevertheless, we can say that the learned reward function R_L satisfies the optimality of the expert's policy π_E since the policy π_L is an optimal policy on the learned reward function R_L and $|V^{\pi_E}(b_0;R_L) - V^{\pi_L}(b_0;R_L)| = 0$. Thus, the reason for our algorithms' failure in the *Rock Sample*[4,3] might be that the objective functions in the optimization problems are not well formulated to choose an appropriate reward function that yields a policy similar to the expert's, among the infinitely many reward functions in the space specified by the constraints of the optimization problems.

We further discuss the details of the results from each problem below. The learned reward functions are compared to the true reward functions for the *Tiger*, *1d Maze*, 5×5 *Grid World*, and *Heaven/Hell* problems, but the reward function in the *Rock Sample*[4,3] problem is omitted since it has too many elements to present.

In the *Tiger* problem, the true and learned reward functions are respectively represented as R^* and R_1 in Table 5. The true reward function is not sparse. Every action is associated with a non-zero reward. Since our methods favor sparse reward functions, there is some degree of difference between the true and the learned reward functions, most notably for the listen action, where our methods assign a zero reward instead of -1 as in the true reward. However, we can apply the policy-invariant reward transformation (Ng et al., 1999) on the learned reward function so that listen action yields -1 reward. R_1 is the transformed learned reward function. It is close to the true reward function and produces the optimal policy whose value is equal to the value of the expert's policy when measured on the true reward function.

For the *1d Maze* problem, the learned reward function is compared to the true reward function in the left panel of Figure 6 and the expert's policy is presented in the right panel of Figure 6. The expert's policy has three nodes: Node n_2 (the starting node) chooses to move right, and changes to node n_1 upon observing *Nothing* or to node n_0 upon observing *Goal*; node n_1 chooses to move right and always changes to node n_0; node n_0 chooses to move left, and changes to node n_2 upon observing *Nothing* or to itself upon observing *Goal*. Following the expert's policy, moving left is

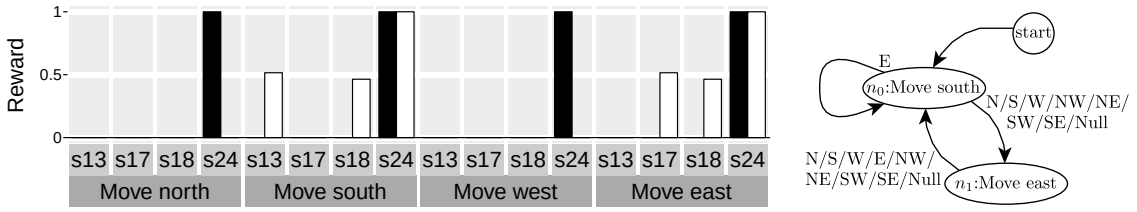

Figure 7: Comparison of the true and the learned reward functions and the expert's policy in the 5×5 *Grid World* problems. *Black bars*: The true reward. *White bars*: The learned reward.

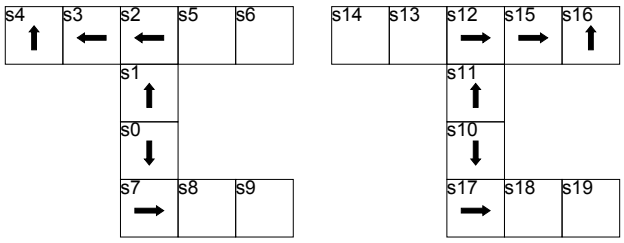

Figure 8: Learned reward function in the *Heaven/Hell* problem. *Black arrow*: $+1$ reward for moving in the direction of the arrow in each state. *Blank grid*: Zero reward for all actions in each state.

always executed after perceiving the goal state. This causes the algorithms to assign the positive reward to moving left in the goal state as the true one, but the zero reward to moving right in the goal state unlike the true one. Consequently, the algorithms find the reward function that explains the behavior of the expert's policy, and the optimal policy from the POMDP with respect to the learned reward function is the same as the expert's policy.

In the 5×5 *Grid World* problem, the expert's policy is simple as depicted in the right panel of Figure 7: The agent alternates moving south and east from the start, visiting the states in the diagonal positions (i.e., $\{s_0, s_5, s_6, s_{11}, s_{12}, s_{17}, s_{18}, s_{23}, s_{24}\}$ and $\{s_0, s_1, s_6, s_7, s_{12}, s_{13}, s_{18}, s_{19}, s_{24}\}$). The learned reward function is presented with the true reward function in the left panel of Figure 7. Our methods assign a small positive reward for moving south in states 13 and 18 and moving east in states 17 and 18. Also, the reward for moving south and east in state 24 is assigned to $+1$ for reaching the goal. The learned reward function closely reflects the behavior of the given expert' policy. Again, even though the learned reward function is different from the true one, it yields the same optimal policy.

Finally, in the *Heaven/Hell* problem, the true reward function is $+1$ for states 4 and 16 being heaven, and -1 for states 6 and 14 being hell. The learned reward is presented in Figure 8 where the agent gets a $+1$ reward when moving in the direction of the arrow in each state. The learned reward function exactly describes the behavior of the expert, which first visits the priest in states 9 and 19 starting from states 0 and 10 to acquire the position of heaven and then moves to heaven in states 4 and 16. As shown in Table 6, the learned reward function in the *Heaven/Hell* problem also yields the policy whose value is equal to that of the expert's policy.

7.2 Experiments on IRL from Sampled Trajectories

The second set of experiments involves the case when the expert's trajectories are given. We experimented on the same set of five problems with all three approaches in Section 6: the max-margin between values (MMV), the max-margin between feature expectations (MMFE), and the projection (PRJ) methods.

In this section, the reward function is assumed to be linearly parameterized with the basis functions and we prepare four sets of basis functions to examine the effect of the choice of basis functions on the performance of the algorithms:

- *Compact*: The set of basis functions that captures the necessary pairs of states and actions to present the structure of the true reward function. Let $F = \{F_0, F_1, \cdots, F_N\}$ be a partition of $S \times A$ such that $\forall (s,a) \in F_i$ have the same reward value $R(s,a)$. The *compact* basis functions for the partition F is defined such that the i-th basis function $\phi_i(s,a) = 1$ if $(s,a) \in F_i$ and $\phi_i(s,a) = 0$ otherwise.

- *Non-compact*: The set of basis functions that includes all the compact basis functions and some extra redundant basis functions. Each basis function ϕ_i is associated with some set of state-action pairs as above.

- *State-wise*: The set of basis functions that consists of the indicator functions for each state. The i-th basis function is defined as $\phi_i(s) = \delta_{s'}(s)$ if i-th state is s'.[7]

- *State-action-wise*: The set of basis functions consists of the indicator functions for each pair of state and action. The i-th basis function is defined as $\phi_i(s,a) = \delta_{(s',a')}(s,a)$ if i-th pair of state and action is (s',a').

For small problems, such as the *Tiger*, *1d Maze*, and 5×5 *Grid World* problems, we experimented with *state-action-wise* basis functions. For the two larger problems, three sets of basis functions are selected. For the *Heaven/Hell* problem, the first set consists of the *compact* set of basis functions. Table 7 shows the set F_i of pairs of states and actions for each basis function. The second set consists of the *state-wise* basis functions and the third set consists of the *state-action-wise* basis functions. For the *Rock Sample*[4,3] problem, the first set consists of the *compact* set of basis functions. The left side of Table 8 shows the set F_i of pairs of states and actions for each basis function. The second set consists of the *non-compact* set of basis functions including the redundant functions that present the rover's using its sensor (ϕ_{10}), moving on the map (ϕ_{11}), sampling at some locations without rocks (ϕ_{12}–ϕ_{15}), and sampling at the rest of the locations (ϕ_{16}). The right side of Table 8 presents the set of the pairs of states and actions for the *non-compact* basis functions. The third set consists of the *state-action-wise* basis functions.

For each experiment, we sampled 2000 belief trajectories. Each trajectory is truncated after a large finite number H of time steps. If we truncate the trajectories after $H_\varepsilon = \log_\gamma(\varepsilon(1-\gamma)/R_{max})$ time steps, the error in estimating the value would be no greater than ε. Table 9 shows the number of time steps for each problem.

As in the previous section, we compare $V^{\pi_L}(b_0; R^*)$ at each iteration, which is the value of the policy π_L from the learned reward function R_L evaluated on the true reward function R^*. The results are shown in Figure 9. All the algorithms found the reward function that generate the policy

7. Here, we use the Kronecker delta function, that is, $\delta_i(j) = 1$ if $i = j$, and $\delta_i(j) = 0$ if $i \neq j$.

F_i	States	Actions
F_0	s_4	$*$
F_1	s_{16}	$*$
F_2	s_6	$*$
F_3	s_{14}	$*$
F_4	$S \backslash \{s_4, s_6, s_{14}, s_{16}\}$	$*$

Table 7: Sets of state-action pairs for the *compact* set of basis functions in the *Heaven/Hell* problem. The states s_4 and s_{16} represent heaven and the states s_6 and s_{14} represent hell.

F_i	States	Actions
F_0	x=0	Move west
F_1	x=3	Move east
F_2	y=0	Move south
F_3	y=3	Move north
F_4	$\langle x,y \rangle = L_0, r_0 = true$	Sample
F_5	$\langle x,y \rangle = L_0, r_0 = false$	Sample
F_6	$\langle x,y \rangle = L_1, r_1 = true$	Sample
F_7	$\langle x,y \rangle = L_1, r_1 = false$	Sample
F_8	$\langle x,y \rangle = L_2, r_2 = true$	Sample
F_9	$\langle x,y \rangle = L_2, r_2 = false$	Sample
F_{10}	$\langle x,y \rangle \notin \{L_i, \forall i\}$	Sample
F_{11}	The remaining state-action pairs	

F_i	States	Actions
$F_0, \cdots, F_9$	Same as in the *compact* set	
F_{10}	$*$	Use the sensor
F_{11}	$*$	Move
F_{12}	$\langle x,y \rangle = L'_0$	Sample
F_{13}	$\langle x,y \rangle = L'_1$	Sample
F_{14}	$\langle x,y \rangle = L'_2$	Sample
F_{15}	$\langle x,y \rangle = L'_3$	Sample
F_{16}	$\langle x,y \rangle \notin \{L_i, \forall i, L'_j, \forall j\}$	Sample
F_{17}	The remaining state-action pairs	

Table 8: Sets of state-action pairs for the *compact* (*Left*) and *non-compact* set of basis functions (*Right*) in the *Rock Sample*[4,3] problem. $\langle x,y \rangle$ denotes the location of the rover. L_i is the location of i-th rock. L'_i is a randomly chosen location without rocks. r_i is the Boolean variable for representing whether the i-th rock is good or not.

close to the expert's policy in small problems, that is, the *Tiger*, *1d Maze*, and 5×5 *Grid World* problems. They also converged to the optimal value in a few iterations when using the *compact* set of basis functions in the two larger problems, that is, the *Heaven/Hell* and *Rock Sample*[4,3] problems. However, more iterations were required to converge when other sets of basis functions were used. This is due to the fact that a larger number of basis functions induces a larger search space. In the *Heaven/Hell* problem, the MMV method converged to a sub-optimal solution using the *state-wise* basis functions although the true reward function can be represented exactly using the *state-wise* basis functions. The MMV method had no such issues when using the *state-action-wise* basis functions. In the *Rock Sample*[4,3] problem, the MMV method also converged to a sub-optimal solution using the *state-action-wise* basis functions with 1024 basis functions, most of them being redundant since there are only 12 basis functions in the *compact* set. Hence, the MMV method is sensitive to the selection of basis functions, whereas the MMFE and PRJ methods robustly yield optimal solutions. Our reasoning on this phenomenon is given in the end of this subsection. Meanwhile, the value of the learned policies tends to oscillate in the beginning of the learning phase, particularly in the *Tiger* and *Rock Sample*[4,3] problems, since our methods are

Problem	# of steps	$V^{\pi_E}(b_0; R^*)$
Tiger	20	1.93
1d Maze	20	1.02
5×5 *Grid World*	50	0.70
Heaven/Hell	300	8.64
Rock Sample[4,3]	200	21.11

Table 9: Configuration for each problem and the value of the expert's policy measured on the true reward function.

Problem	ϕ	$\lvert\phi\rvert$	$V^{\pi_L}(b_0; R^*)$			Time		
			MMV	MMFE	PRJ	MMV	MMFE	PRJ
Tiger	SA	6	1.79	1.93	1.93	10.04 (72.27)	7.04 (41.56)	3.97 (96.33)
1d Maze	SA	8	1.02	1.02	1.02	0.88 (75.07)	5.18 (10.83)	0.71 (82.13)
5×5 *Grid World*	SA	100	0.70	0.70	0.70	24.10 (95.11)	20.07 (96.88)	21.49 (98.16)
Heaven/Hell	C	5	8.49	8.64	8.64	18.54 (63.02)	11.80 (79.75)	8.99 (88.66)
	S	20	5.70	8.64	8.64	375.03 (96.48)	332.97 (98.51)	937.59 (99.75)
	SA	80	8.47	8.64	8.64	443.57 (98.31)	727.87 (99.30)	826.37 (99.68)
Rock Sample[4,3]	C	11	20.84	20.05	20.38	8461.65 (99.16)	8530.18 (52.03)	10399.61 (59.86)
	NC	17	20.83	20.62	20.16	21438.83 (89.88)	10968.81 (25.05)	27808.79 (79.41)
	SA	1024	-26.42	17.83	19.05	31228.85 (72.45)	13486.41 (78.25)	16351.59 (80.57)

Table 10: Results of IRL for POMDP\R from sampled trajectories. The sets of the basis functions are denoted by C(*compact*), NC(*non-compact*), S(*state-wise*), and SA(*state-action-wise*). The average computation time for each trial is reported in seconds and the numbers in the parentheses next to the computation time are the percentages of the time taken by the POMDP solver.

not guaranteed to improve monotonically and are hence prone to yielding poor intermediate reward functions. However, these poor intermediate reward functions will effectively restrict the region of the reward functions for the final result.

We summarize $V^{\pi_L}(b_0; R^*)$ returned at the end of the algorithms and the computation time for each trials with the computation time for solving intermediate POMDPs in Table 10. As noted in the above, in most of the experiments, the algorithms eventually found the policy whose performance is the same as the expert's, which means the algorithms found the reward function that successfully recovers the expert's policy. The computation time increased when the size of basis functions and the size of the problems were increased. When the *state-action-wise* basis functions were applied for the *Rock Sample*[4,3] problem, it took about 8 hours on average for the MMV method to converge. However, the larger portion of the computation time was spent for solving intermediate POMDPs. The average percentage of the time spent for solving intermediate POMDPs was 78.83%.

The third set of experiments was conducted for examining the performance of the algorithms as the number of sampled belief trajectories varied. We experimented with the MMV, MMFE, and PRJ methods in the *Tiger* problem. Each trajectory was truncated after 20 time steps. Figure 10 presents the results where the value of policy is measured by $V^{\pi_L}(b_0; R^*)$. The MMFE method required fewer number of trajectories to attain the policy that performs close to the expert's than the MMV and PRJ methods required. The performance of the PRJ method was the worst when given

few sampled trajectories, but it improved fast as the number of trajectories increased. However, the MMV method needed many trajectories to find the near-optimal solution.

We conclude this subsection with our reasoning on why the MMFE and PRJ methods typically outperform the MMV method. The MMFE and PRJ methods directly use the differences in feature expectations (line 11 in Algorithm 5 and line 14 in Algorithm 6), whereas the MMV method uses the differences in values obtained from the *weight vectors* and feature expectations (line 12 in Algorithm 4). Using the differences in values can be problematic because it is often possible that a weight vector very different from the true one can yield a very small difference in values. Hence, it is preferable to directly use the differences in feature expectations since it still bounds the differences in values without depending on the weight vectors.

8. Related Work

In control theory, recovering a reward function from demonstrations has received significant attention, and has been referred to as the inverse optimal control (IOC) problem. It was first proposed and studied for linear systems by Kalman (1964). IRL is closely related to IOC, but the focus is on the problem of inverse optimality within the framework of RL. As already mentioned in the introduction, Russell (1998) proposed IRL as an important problem in machine learning, suggesting that it will be useful in many research areas such as studies on animal and human behaviors since the reward function reflects the objective and the preference of the decision maker. IRL is also useful for reinforcement learning since similar but different domains often share the same reward function structure albeit different dynamics. In this case, transferring the reward function learned from one domain to another domain may be useful.

Besides the task of *reward learning*, IRL has gained interest in *apprenticeship learning*, where the task is to find the policy with possibly better performance than the one demonstrated by an expert. Apprenticeship learning is useful when explicitly specifying the reward function is difficult but the expert's behaviors are available instead. Apprenticeship learning is a promising approach in robotics since it provides a framework for a robot to imitate the demonstrator without a full specification of which states are good or bad, and to what degree.

Since Russell (1998), a number of algorithms for IRL and apprenticeship learning have been proposed in the last decade. Most of the algorithms assume a completely observable setting, where the agent has capability to access the true global state of the environment often modeled as an MDP. In this section, we briefly review some of these previous works on the IRL and apprenticeship learning problem.

One of the first approaches to IRL in the MDP setting was proposed by Ng and Russell (2000), which we have covered in Section 3. They presented a sufficient and necessary condition on the reward functions which guarantees the optimality of the expert's policy, and provided some heuristics to choose a reward function since the degenerate reward functions also satisfy the optimality condition. The IRL problem was formulated as LP with the constraints corresponding to the optimality condition and the objective function corresponding to the heuristics. The algorithm was shown to produce reasonably good solutions in the experiments on some benchmark problems. We have extended this algorithm to the partially observable setting in Section 5 and Section 6.1.

Abbeel and Ng (2004) presented an apprenticeship learning algorithm based on IRL, which we have described in Section 3.2. One of the important aspects of the algorithm was to compare the feature expectations between the expert's and the learned policies rather than the estimated values.

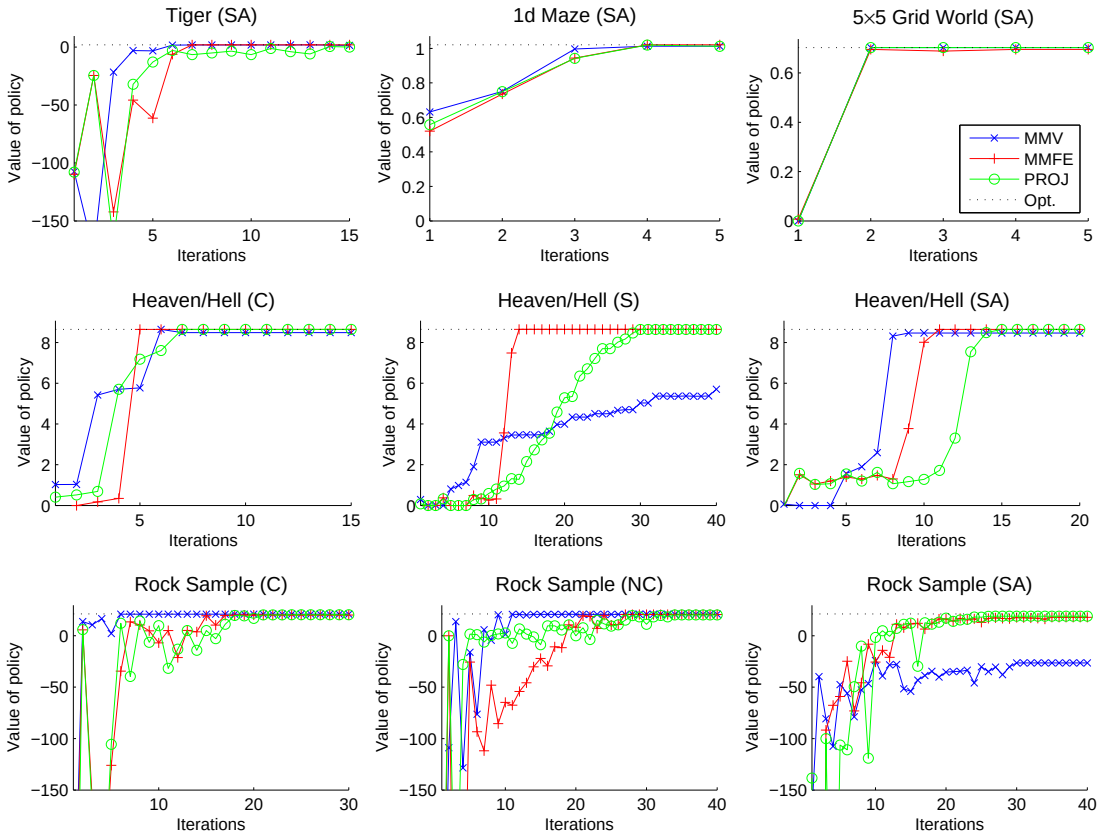

Figure 9: The value of the policies produced by the learned reward function at each iteration by the algorithms of IRL for POMDP\R from sampled trajectories. The value is measured on the true reward function for each problem. The optimal value is denoted by *Opt.* in the legend.

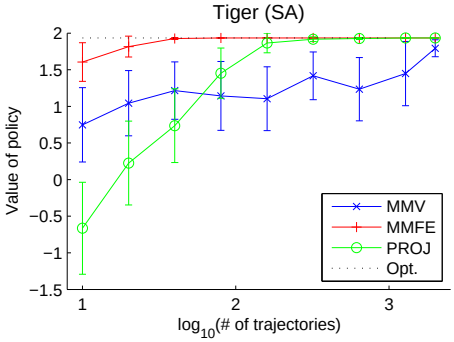

Figure 10: The value of the policies produced by the learned reward function by the algorithms of IRL for POMDP\R from varying number of sampled trajectories. Averages over 100 trials are presented with 95% confidence intervals. The x-axis represents the number of sampled trajectories on a log 10 scale.

The algorithm comes with a theoretical guarantee that the learned policy is similar to the expert's policy when evaluated on the true reward function. The algorithm was shown to successfully learn different driving styles in a simulated car driving task. This work was further extended using a number of different approaches. We have extended this algorithm to the partially observable setting in Section 6.2 and Section 6.3.

The structured max-margin optimization technique (Taskar et al., 2005) was applied to apprenticeship learning by Ratliff et al. (2006). They formulated a QP problem to find the weight vector of the reward basis functions that maximizes the margin between the expert's policy and all other policies. They also provided the maximum margin planning (MMP) algorithm based on the subgradient method, which is faster than the QP method. The MMP was shown to solve problems of practical sizes, such as route planning for outdoor mobile robots, where the QP method was not applicable.

Neu and Szepesvari (2007) proposed an algorithm for apprenticeship learning that unifies the direct and indirect methods: The direct method, using supervised learning methods, finds the policy that minimizes loss functions that penalize deviating from the expert's policy. The indirect method finds the policy using the learned reward function from IRL. Since the loss functions are defined on the policy space, the algorithm uses natural gradients to map the gradients in the policy space to those in the weight vector space of reward functions.

Whereas most of the apprenticeship learning algorithms focus on approximating the performance of the expert's policy, Syed and Schapire (2008) proposed a method called multiplicative weights for apprenticeship learning (MWAL), which tries to improve on the expert's policy. This was achieved in a game-theoretic framework using a two person zero-sum game, where the learner selects a policy that maximizes its performance relative to the expert's and the environment adversarially selects a reward function that minimizes the performance of the learned policy. The game was solved using the multiplicative weights algorithm (Freund and Schapire, 1999) for finding approximately optimal strategies in zero-sum games.

One of the difficulties in apprenticeship learning is that most proposed algorithms involve solving MDPs in each iteration. Syed et al. (2008) addressed this issue by identifying the optimization performed in the MWAL algorithm, and formulating it into an LP problem. They showed that this direct optimization approach using an off-the-shelf LP solver significantly improves the performance in terms of running time over the MWAL algorithm.

As mentioned in Section 4, IRL is an ill-posed problem since the solution of IRL is not unique. To address the non-uniqueness in the solution, the above approaches adopt some heuristics, for example, maximizing the margin between the expert's policy and other policies. We could also handle the uncertainty in the reward function using probabilistic frameworks. Ramachandran and Amir (2007) suggested a Bayesian framework for IRL and apprenticeship learning. The external knowledge about the reward function is formulated in the prior, and the posterior is computed by updating the prior using the expert's behavior data as evidence. Ziebart et al. (2008) proposed an apprenticeship learning algorithm adopting the maximum entropy principle for choosing the learned policy constrained to match feature expectations of the expert's behavior.

Recently, Neu and Szepesvari (2009) provided a unified framework for interpreting a number of incremental IRL algorithms listed above, and discussed the similarities and differences among the algorithms by defining the distance function and the update step employed in each algorithm. Each algorithm was characterized by the distance function that measures the difference between the expert's behavior data and the policy from the learned reward function, and the update step that computes new parameter values for the reward function.

The question of whether the IRL and the apprenticeship learning algorithms listed above can be extended to the partially observable setting in an efficient way remains as an important open problem.

9. Conclusion

The objective of IRL is to find the reward function that the domain expert is optimizing from the given data of her or his behavior and the model of the environment. IRL will be useful in various areas connected with reinforcement learning such as animal and human behavior studies, econometrics, and intelligent agents. However, the applicability of IRL has been limited since most of the previous approaches employed the assumption of an omniscient agent using the MDP framework.

We presented an IRL framework for dealing with partially observable environments in order to relax the assumption of an omniscient agent in the previous IRL algorithms. First, we derived the constraints of the reward function to guarantee the optimality of the expert's policy and built optimization problems to solve IRL for POMDP\R when the expert's policy is explicitly given. The results from the classical POMDP research, such as the generalized Howard's policy improvement theorem (Howard, 1960) and the witness theorem (Kaelbling et al., 1998), were exploited to reduce the computational complexity of the algorithms. Second, we proposed iterative algorithms of IRL for POMDP\R from the expert's trajectories. We proposed an algorithm that uses max-margin between values via LP, and then, in order to address larger problems robustly, we adapted the algorithms for apprenticeship learning in the MDP framework to IRL for POMDP\R. Experimental results on several POMDP benchmark domains showed that, in most cases, our algorithms robustly find solutions close to the true reward function, generating policies that acquire values close to that of the expert's policy.

We demonstrated that the classical IRL algorithm on MDP\R could be extended to POMDP\R, and we believe that more recent IRL techniques as well as some of the IRL-based apprenticeship learning techniques could be similarly extended by following our line of thought. However, there are a number of interesting issues that should be addressed in future studies.

9.1 Finding the Optimality Condition

The proposed conditions in Section 5 are not sufficient conditions of the reward function to guarantee the optimality of the expert's policy. The condition based on the comparison of Q-functions in Equation (14) should be evaluated for every possible policy that may have an infinite number of nodes. The condition using the DP update and the witness theorem in Equation (16) should be evaluated for some useful nodes that the expert's policy may not have due to their unreachability from the starting node. Also, Equations (14) and (16) should be extended to assess the value for all beliefs. Thus, it is crucial to find a sufficient condition that can be efficiently computed in order to restrict the feasible region of the reward functions tightly so that the optimization problems can find the reward function that guarantees the optimality of the given expert's policy.

9.2 Building an Effective Heuristic

Although the constraints for the reward function are not sufficient conditions, we empirically showed that $|V^{\pi_E}(b_0;R_L) - V^{\pi_L}(b_0;R_L)| = 0$, which implies that the value of the expert's policy π_E is equal to that of the optimal policy π_L produced by the learned reward R_L when the value is evaluated on

the learned reward. In other words, the expert's policy is another optimal policy for the learned reward and the learned reward still satisfies the optimality condition of the expert's policy. However, the optimal policy for the learned reward does not achieve the same value as the expert's policy when the value is evaluated on the true reward. The reason for the algorithms' failure to find the appropriate reward function may lie in the shortcomings of the heuristic for the objective functions. In this paper, we use the heuristic originally proposed by Ng and Russell (2000). It prefers the reward function that maximizes the sum of the differences between the value of the expert's policy and the other policies while forcing the reward function to be as sparse as possible. Unfortunately, this heuristic failed in some cases in our experiments. Hence, a more effective heuristic should be devised to find the reward function that provides similar behavior to the expert's policy. This can be addressed by adapting more recent IRL approaches such as the Bayesian IRL (Ramachandran and Amir, 2007) and the maximum entropy IRL (Ziebart et al., 2008) to partially observable environments. The Bayesian IRL prefers the reward function inducing the high probability of executing actions in the given behavior data, and the maximum entropy IRL prefers the reward function maximizing the entropy of the distribution over behaviors while matching the feature expectations.

9.3 Scalability

The algorithms we presented are categorized into two sets: The first is for the cases when the expert's policy is explicitly given in the FSC representations and the second is for the cases when the trajectories of the expert's executed actions and the corresponding observations are given. For the first set of the algorithms, the computational complexity is reduced based on the generalized Howard's policy improvement theorem (Howard, 1960) and the witness theorem (Kaelbling et al., 1998). The algorithms still suffer from a huge number of constraints in the optimization problem. The question is then whether it is possible to select a more compact set of constraints that define the valid region of the reward function while guaranteeing the optimality of the expert's policy, which is again related to finding the sufficient condition. For the second set of the algorithms, the scalability is more affected by the efficiency of the POMDP solver than by the number of constraints in the optimization problem. Although PBPI (Ji et al., 2007), the POMDP solver used in this paper, is known to be one of the fastest POMDP solvers which return FSC policies, it was observed in the experiments that the algorithms spent more than 95% of the time to solve the intermediate POMDP problems. Computing an optimal policy in the intermediate POMDP problem takes a much longer time than solving a usual POMDP problem, since an optimal policy of the intermediate POMDP problem is often complex due to the complex reward structure. The limitation could be handled by modifying the algorithms to address the IRL problems with other POMDP solvers, such as HSVI (Smith and Simmons, 2005), Perseus (Spaan and Vlassis, 2005), PBVI (Pineau et al., 2006), and SARSOP (Kurniawati et al., 2008), which generate the policy defined as a mapping from beliefs to actions.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant 2009-0069702, and by the Defense Acquisition Program Administration and Agency for Defense Development of Korea under contract 09-01-03-04.

References

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pages 1–8, Banff, Alta, Canada, 2004.

Tilman Borgers and Rajiv Sarin. Naive reinforcement learning with endogenous aspirations. *International Economic Review*, 41(4):921–950, 2000.

Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI)*, pages 1023–1028, Seattle, WA, USA, 1994.

Jaedeug Choi and Kee-Eung Kim. Inverse reinforecement learning in partially observable environments. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1028–1033, Pasadena, CA, USA, 2009.

Michael X Cohen and Charan Ranganath. Reinforcement learning signals predict future decisions. *Journal of Neuroscience*, 27(2):371–378, 2007.

Ido Erev and Alvin E. Roth. Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *American Economic Review*, 88(4): 848–881, 1998.

Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1–2):79–103, 1999.

Hector Geffner and Blai Bonet. Solving large POMDPs using real time dynamic programming. In *Proceedings of the AAAI Fall Symposium Series*, pages 61–68, 1998.

Jacques Hadamard. Sur les problemes aux derivees partielles et leur signification physique. *Princeton University Bulletin*, 13(1):49–52, 1902.

Eric A. Hansen. *Finite-Memory Control of Partially Observable Systems*. PhD thesis, University of Massachusetts Amherst, 1998.

Jesse Hoey, Axel Von Bertoldi, Pascal Poupart, and Alex Mihailidis. Assisting persons with dementia during handwashing using a partially observable Markov decision process. In *Proceedings of the 5th International Conference on Vision Systems*, Bielefeld University, Germany, 2007.

Ed Hopkins. Adaptive learning models of consumer behavior. *Journal of Economic Behavior and Organization*, 64(3–4):348–368, 2007.

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

Shihao Ji, Ronald Parr, Hui Li, Xuejun Liao, and Lawrence Carin. Point-based policy iteration. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1243–1249, Vancouver, BC, USA, 2007.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

Rudolf E. Kalman. When is a linear control system optimal? *Transaction ASME, Journal Basic Engineering*, 86:51–60, 1964.

Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of Robotics: Science and Systems*, Zurich, Switzerland, 2008.

Terran Lane and Carla E. Brodley. An empirical study of two approaches to sequence learning for anomaly detection. *Machine Learning*, 51(1):73–107, 2003.

Daeyeol Lee, Michelle L. Conroy, Benjamin P. McGreevy, and Dominic J. Barraclough. Reinforcement learning and decision making in monkeys during a competitive game. *Cognitive Brain Research*, 22(1):45–58, 2004.

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

P. Read Montague and Gregory S. Berns. Neural economics and the biological substrates of valuation. *Neuron*, 36(2):265–284, 2002.

Gergely Neu and Csaba Szepesvari. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 295–302, Vancouver, BC, Canada, 2007.

Gergely Neu and Csaba Szepesvari. Training parsers by inverse reinforcement learning. *Machine Learning*, pages 1–35, 2009.

Allen Newell. The knowledge level. *Artificial Intelligence*, 18(1):87–127, 1982.

Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pages 663–670, Stanford University, Standord, CA, USA, 2000.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, pages 278–287, Bled, Slovenia, 1999.

Yael Niv. Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3):139–154, 2009.

Joelle Pineau, Geoffrey Gordon, and Sebastian Thrun. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380, 2006.

Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *Proceedings of 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2586–2591, Hyderabad, India, 2007.

Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 729–736, Pittburgh, Pennsylvania, USA, 2006.

Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)*, pages 101–103, Madison, WI, USA, 1998.

Trey Smith. *Probabilistic planning for robotic exploration*. PhD thesis, Carnegie Mellon University, The Robotics Institute, 2007.

Trey Smith and Reid Simmons. Heuristic search value iteration for POMDPs. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 520–527, Banff, Canada, 2004.

Trey Smith and Reid Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 542–547, Edinburgh, Scotland, 2005.

Edward Jay Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, 1971.

Matthijs T. J. Spaan and Nikos Vlassis. A point-based pomdp algorithm for robot planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2399–2404, New Orleans, LA, USA, 2004.

Matthijs T. J. Spaan and Nikos Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

Umar Syed and Robert E. Schapire. A game-theoretic approach to apprenticeship learning. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 1449–1456, Vancouver, British Columbia, Canada, 2008.

Umar Syed, Michael Bowling, and Robert Schapire. Apprenticeship learning using linear programming. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1032–1039, Helsinki, Finland, 2008.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: a large margin approach. In *Proceedings of the 22nd International Conference on Machine learning*, pages 896–903, Bonn, Germany, 2005.

Jason D. Williams and Steve Young. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422, 2007.

Qing Zhao, Lang Tong, Ananthram Swami, and Yunxia Chen. Decentralized cognitive mac for opportunistic spectrum access in ad hoc networks: A pomdp framework. *IEEE Journal on Selected Areas in Communications*, 25(3):589–600, 2007.

Brian D. Ziebart, Andrew Maas, James Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1433–1438, Chicago, IL, USA, 2008.