Uncertain Reward-Transition MDPs for Negotiable Reinforcement Learning

Nishant Desai

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-231
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-231.html

December 16, 2017

Uncertain Reward-Transition MDPs for Negotiable Reinforcement Learning

by

Nishant Desai

A thesis submitted in partial satisfaction of the
requirements for the degree of
Masters of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Fall 2017

Abstract

Uncertain Reward-Transition MDPs for Negotiable Reinforcement Learning

by

Nishant Desai

Masters of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Markov Decision Processes (MDPs) allow us to build policies for maximizing the expectation of an objective in stochastic environments where the state of the world is fully observed. Partially Observable MDPs (POMDPs) give us machinery for planning when we have additional uncertainty over the state of the world. We can make a similar jump from MDPs to characterize uncertainty over other elements of the environment. Namely, we can also have uncertainty over the transition and reward functions in an MDP. Here, we introduce new classes of uncertain MDPs for dealing with these kinds of uncertainty and present several folk theorems showing that certain subsets of these can be reduced to the standard POMDP with uncertainty only over the state. We are particularly interested in developing these frameworks to explore applications to Negotiable Reinforcement Learning, a method for dynamically balancing the utilities of multiple actors.

Contents

List of Figures

Acknowledgments

I would like to thank my advisor, Stuart Russell. When ideas abound, he is an expert at separating the wheat from the chaff.

I'd also like to thank Andrew Critch, my closest collaborator on the present work and the source of many of the ideas underlying this thesis. Dmitrii Krasheninov, during his summer in Berkeley, also contributed greatly to this work. Thanks to Peter Bartlett, my second reader, who provided thoughtful feedback on this report.

I was fortunate to have the opportunity to work on a number of other research projects during my short program. I extend my thanks to my collaborators on those projects: Siddharth Srivastava, Rick Freedman, Shlomo Zilberstein, Lisa Goldberg, and Daniel Filan. Thanks to all of RUGS and CHAI for many stimulating conversations over the years.

I am thankful to Alon Daks, who has been my closest collaborator since our time as Freshman computer science students, and much of my academic success can be attributed to our time spent together. I am thankful to Gunjan Baid: without her support none of this would have happened. I am thankful to my parents for their unending encouragement and commitment to my education.

Finally, I would be remiss if I did not acknowledge Caffe Strada, my home away from home.

Chapter 1

Introduction

The recently developed negotiable reinforcement learning (NRL) framework seeks to answer the question of how an expected-utility-maximizing agent should behave when it is simultaneously owned by two agents with differing utility functions and differing beliefs about the transition model of the environment [3]. The problem is motivated by the idea that at some point in the future, artificial agents will have to simultaneously answer to multiple owners: a household robot may have to consider the needs of an entire family, corporations may enter into joint ventures to build decision agents the way they presently do for capital infrastructure, and many scenarios yet to be conceived will likely involve a single agent working for multiple owners. In each case, these parties will only enter into a co-ownership agreement if each actor believes that the jointly owned agent will act in a way beneficial to them.[1] Therefore, the idea of Pareto optimality gives us a minimum condition that must be met by any sufficiently capable AI: a utility balancing mechanism is not worth much if all parties can be better off under some other scheme.

The purpose of the NRL framework (pronounced "nurl") is to describe a Pareto optimal scheme for balancing these utilities. A simple scheme for accomplishing this balance is to optimize for a weighted sum of the actors' utilities. However, this turns out not to be Pareto

[1]The problem is also of interest to members of the AI safety community who believe that future AGI will likely be owned by a coalition of powerful actore (e.g. national governments).

optimal when the players have different beliefs about the environment. By assumption, these actors would engage in negotiation about the behavior of the agent, and it is plausible that the differences under negotiation will arise from differing beliefs about the transition model of the world. Instead of the simple weighting scheme, the NRL framework proposes a utility balancing scheme that resembles bet settling: the agent weights each player's utility proportional to the extent to which that player's stated belief aligns with the agent's observations.

In the present work, we take steps towards implementing a NRL agent. To accomplish this, we must first develop a framework that allows us to connect the NRL problem to existing algorithms in the AI literature. To that end, we present a model that we call an uncertain reward-transition MDP (URTMDP). Partially observable Markov decision processes (POMDPs) are a well studied model for planning in an environment with Markovian transitions for an expected-utility-maximizing agent with uncertainty about its state. URTMDPs model planning under uncertainty about the transition and reward function components of an MDP. This model can be further generalized into the concept of a U*MDP, where the * can be replaced with any combination of reward function, state, and transition model uncertainty. As our main result, we present a folk theorem about this generalized uncertainty model stating that it can be reduced to a POMDP and solved using existing algorithms for planning under state uncertainty.

We show that the NRL problem can be described by a URTMDP. In doing so, we admit the use of well understood POMDP algorithms in solving the problem of balancing opposing utility functions. We use this observation to reduce a NRL game to a POMDP and use point-based value iteration (PBVI) to learn a policy for a NRL agent in a simple grid-world environment. We then observe the behavior of this agent and discuss the ways in which it trades off between agent utilities as a function of its observations. We also analyze how each player, passively observing the NRL agent, evaluates its behavior. This is of particular interest, as the guarantee of Pareto optimality may bring opposing actors to the negotiating table, but if players feel the NRL agent's performance is not up to par, they may feel compelled to break the agreement in favor of building a fully-owned artificial agent.

1.1 Outline

The remainder of this report has the following structure. Chapter 2 presents background information on the POMDP model and algorithms for solving POMDPs, and it presents a deeper background on the NRL framework. Chapter 3 develops the U*MDP framework and presents the reduction from URTMDP to POMDP. Chapter 4 discusses the details of implementing a NRL agent and discusses experimental results about the agent. Chapter 5 presents related work, discussing models elsewhere in the AI literature that can be described in the U*MDP framework. Chapter 6 concludes the report and outlines future avenues of research following from the present work.

Chapter 2

Background

This chapter provides background information on algorithms and frameworks used in this work and introduces the negotiable reinforcement learning (NRL) problem that will be the primary application explored in later chapters. In the first section, we discuss the partially observable Markov decision process (POMDP) framework for decision making under uncertainty. In the second section, we discuss techniques for approximating optimal policies in POMDPs. In the third section, we discuss the framework of negotiable reinforcement learning.

2.1 Partially Observable Markov Decision Processes

The partially observed Markov decision process is a framework for modelling planning under uncertainty. The framework assumes an agent acting in an environment with Markovian dynamics where the agent is unable to directly observe its state [19]. Instead, the agent receives observations from the environment and, by assumption, knows the conditional probability of receiving each possible observation in each possible state. Included in the model is a reward function that assigns a score to the agent's decisions at each timestep. The agent's goal is to choose actions to maximize its expected reward over time.

Formally, a POMDP can be expressed as a tuple $\langle S, A, T, R, \Omega, O\rangle$ [10]:

- S is a set of possible environment states.
- A is the decision agent's action space: the set of possible actions the agent can take in each state.
- $T : S \times A \mapsto \Delta S$ is a transition model. For any given state, s, and action, a, and subsequent state, s' it gives the probability of transitioning to state s' after taking action a in state s. ΔS here is the set of all probability distributions over S.
- $R : S \times A \mapsto \mathbb{R}$ is the reward signal. A POMDP agent's goal is to select actions that maximize the expected sum of this reward over time.
- Ω is a set of possible observations.
- $O : S \times A \mapsto \Delta\Omega$ is the observation function. For any state, action, observation tuple $\langle s', a, \omega\rangle$, O gives the probability that the agent will receive observation ω after taking action a and transitioning into state s'.

Because the agent's reward is a function of the latent state, the agent must maintain a belief state, $b \in \Delta S$, a probability distribution over the state space S. For any state s, the probability assigned by b to s represents the agent's subjective belief that it is in state s. As the agent takes actions and receives observations, it uses Bayes' Rule to update its belief [15]. Let b' be the updated belief state. After taking an action, a, and receiving an observation, ω, the agent's new belief is computed as:

$$b'(s') \propto \sum_{s\in S} P(\omega|s')P(s'|s,a)b(s).$$

The belief state and its update rule capture all of the uncertainty in a POMDP. The agent's policy π, the function that determines which action the agent chooses at each timestep, only needs to depend on its belief state, which is fully observed by the agent. In fact, finding the reward-maximizing policy π^* for a POMDP can be reduced to finding the optimal policy for a fully-observed Markov decision process on the belief state [15].

2.2 Solving POMDPs

An algorithm for solving POMDPs must return a policy, $\pi : \Delta S \mapsto A$, which is a function from the belief state to an action.[1] This policy determines the agent's behavior: at each timestep, the current belief is used to determine the agent's action.

The optimal policy, π^*, is the policy that maximizes the expected discounted sum of rewards. We can express this condition as:

$$\pi^*(b) = \arg\max_{\pi \in \Pi} \mathbb{E}\left[\sum_t \sum_{s \in S} \sum_{s' \in S} \gamma^t b_t(s) T(s, \pi(b_t), s') R(s', \pi^*(s'))\right], \tag{2.1}$$

where $\gamma \in (0, 1]$ is a discount factor [14].

Since POMDPs can be reduced to MDPs on the belief state, it is possible, in principle, to use the value iteration algorithm [15] to solve for π^*. However, this would require iterating over all possible belief states. The number of belief states is proportional to the number of unique observation histories to plan over, and with stochastic dynamics and noisy observations the number of histories becomes intractably large even for very simple problems. As a result, many approaches to planning policies for POMDPs rely on point-based approximations to value iteration.

All point-based approximations rely on estimating the value of the optimal policy. In sequential decision processes, the value of a fixed policy π in a state s is defined as the expected sum of all future rewards starting from s and following π. The state-action value function is defined counterfactually as the expected sum of future rewards gained by taking some action a in state s and following policy π for all future timesteps. In a POMDP, the state-action value function is linear in the belief state and is parametrized by a vector α [15]. Roughly speaking, the optimal policy is found by taking the supremum over all α at every point in the belief space and following the corresponding action at each belief point. Point-based algorithms attempt to estimate this value manifold by sampling belief points and

[1] Because the belief state can be computed using the initial belief b_0, which is part of the model, and the action-observation history, $h_t = \langle a_0, \omega_0, \ldots, a_{t-1}, \omega_{t-1}\rangle$, we can also write the policy as a function of the history $\pi(h_t)$. We will adopt this notation moving forward.

computing approximate state-action value vectors on these sampled states. The operator used to compute this approximation is called the point-based backup operator [12].

The simplest of these algorithms is called point-based value iteration (PBVI) [12]. PBVI constructs its belief set by simulating trajectories in the POMDP and iteratively expanding the set of belief points by adding in new belief states that are far from the current set. Other point-based planning algorithms make use of the same backup operator but use more sophisticated procedures for choosing the belief set (e.g. [17], [20]). For simplicity, we will use PBVI for our later experiments.

A number of online approaches also exist for learning policies for POMDPs. These approaches tend to use tree-search to estimate values and learn better estimates as the agent observes more trajectories. POMCP, a relatively recent algorithm that has proven to be successful on large scale POMDPs, is based on a tree search that estimates values by taking random trajectories [16]. These algorithms may serve as the basis for more efficient NRL implementations in the future.

A thorough overview of the POMDP framework and algorithms can be found in [14].

2.3 Negotiable Reinforcement Learning

The negotiable reinforcement learning (NRL) framework describes a Pareto optimal scheme for balancing the utility functions of players that have differing beliefs about the environment [3]. The framework models the situation where two agents with differing utility functions and beliefs desire to jointly own a third utility-maximizing agent.

The question arises of what the utility function of the co-owned agent should be. The economics literature suggests that for agents with utility functions u_1 and u_2, the aggregate utility should be expressed as a weighted sum of these two:

$$u = w_1 u_1 + w_2 u_2,$$

where the weights w_1 and w_2 are decided upon by the players at the time the co-owned agent is created [9]. However, offline planning algorithms require models of dynamics and state space as inputs. In the case where the players cannot agree on these model parameters, planning cannot be executed for the joint agent to compute a policy. Furthermore, as [3] shows, this scheme is not Pareto optimal in the case of a sequential decision problem where the two players have differing beliefs about the transition model and state space of the environment. This is roughly due to the fact that each player assesses the probability distributions in the expected reward calculation shown in Equation 2.1 differently.

In this setting, the following scheme turns out to be Pareto optimal. By assumption, the players agree on the action space A and the observation space Ω of their co-owned agent. First, each player specifies their beliefs as a model that assigns a likelihood to any observation history, conditioned on actions. We refer to these beliefs as $P_1(h_t)$ and $P_2(h_t)$. At each timestep, the policy selects actions according to the following weighted expected utility calculation:

$$\pi(h_t) = \arg\max_{a_t \in A} \Big(w_1 P_1(h_t) \mathbb{E}[U_1 | a_t, h_t, \pi] + w_2 P_2(h_t) \mathbb{E}[U_2 | a_t, h_t, \pi] \Big)$$

where the expectations are calculated using Equation 3.2 with respect to the appropriate probability distributions. In words, at each timestep, an action is chosen to maximize a weighted sum of future utilities, where the weight is given by the posterior likelihood assigned to h_t by each player's belief.[2] This is analogous to maintaining a belief state about the true transition model of the environment. We will explore this connection in more depth later on.

[2]In the case that players share the same beliefs, the policy computed in this way is equivalent to a policy computed using Equation 2.3 as a utility function [3].

Chapter 3

Framework

In this chapter, we present the theoretical framework we will be using to explore negotiable reinforcement learning and general problems of unknown transition model. Section 3.1 presents the Uncertain Reward-Transition MDP, which will be the class of decision problem associated with NRL problems. Section 3.2 generalizes the framework of Section 3.1 to other types of uncertainty in decision problems.

3.1 Uncertain Reward-Transition MDP

This section introduces the Uncertain Reward-Transition MDP, an MDP where the agent does not know the transition model or reward function and has a joint belief over the transition function, T, and the reward function, R, both of which are treated as random variables. The fact that this belief is a joint distribution over $\Delta(\boldsymbol{\mathcal{R}} \times \boldsymbol{\mathcal{T}})$ is important for the bet-settling aspects of negotiable reinforcement learning. If the distribution is chosen such that R and T are not independent, then learning about the transition model gives information about the reward function. We elaborate on this observation later on.

Definition 3.1.1. *An Uncertain Reward-Transition MDP (URTMDP) is a sequential decision problem in which the agent can observe its state but does not have direct access to the* *transition model of the environment or its reward function. A URTMDP, M, consists of a tuple $\langle S, s_0, A, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{R}}, b_{RT} \rangle$. The elements of this tuple are defined as follows:*

- *S: The set of possible environment states, s.*
- *s_0: The the initial state distribution, $s_0 \in \Delta S$.*
- *A: The set of possible agent actions, a.*
- *$\boldsymbol{\mathcal{T}}$: A set of possible transition models, $T : S \times A \mapsto S$.*
- *$\boldsymbol{\mathcal{R}}$: A set of possible reward functions, $R : S \times A \mapsto \mathbb{R}$.*
- *b_{RT}: The agent's initial belief about T and R. b_{RT} is a probability distribution over transition and reward functions, $b_{RT} \in \Delta(\boldsymbol{\mathcal{R}} \times \boldsymbol{\mathcal{T}})$.*

T is not known to the agent, it is able to observe the results of state transitions. Using this information, along with its prior over the transition model, the agent can compute a posterior likelihood, $P(T|s, a, s')$, for each $T \in \boldsymbol{\mathcal{T}}$ and for every state transition observation (s, a, s'). The result of this Bayesian updating is that the agent is able to learn T over time by taking actions in the environment.

In contrast, the agent does not receive any information about its reward function. The reward is a function of the environment state computed by the agent itself, not a signal exogenous to the agent. As a result, the posterior likelihood of a given reward function, $P(R|s, a, s')$, depends only on the agent's posterior belief about T. To see this, note that

$$P(R|s, a, s') = \sum_{T \in \boldsymbol{\mathcal{T}}} P(R|T, s, a, s') P(T|s, a, s'). \tag{3.1}$$

Given T, the observed transition tuple carries no additional information about R, so we get that $P(R|T, s, a, s') = P(R|T)$. The posterior reward likelihood depends only on the posterior transition likelihood and the prior conditional probability of a reward function given T.

Therefore, an expected-reward-maximizing agent always acts as though its reward, conditioned on T, is known and equal to the conditionally expected reward function. We prove this in Theorem 3.1.2 as a step towards reducing the URTMDP to a POMDP. In this proof, we consider only the case of a UTRMDP with discrete state and action spaces, S and A, and a discrete initial belief distribution b_{RT}. However, an analogous proof is possible in the case of continuous state and action spaces and a continuous belief distribution.

Theorem 3.1.2. *The optimal policy, π^* for any URTMDP, M, is also the optimal policy for some URTMDP, M', where $P_{b_{RT}}(R|T)$ has support over only one element of $\boldsymbol{\mathcal{R}}$ for each $T \in \boldsymbol{\mathcal{T}}$.*

Proof. The optimal policy, π^*, for a URTMDP, M, selects actions to maximize the expected sum of discounted rewards over time. This expectation is given by the following expression:

$$\begin{aligned}\mathop{\mathbb{E}}_{h_t,T,R}\left[\sum_t \gamma^t R(s_t,\pi^*(h_t))\right] &= \sum_t \sum_{s_t\in S} \sum_{T\in\boldsymbol{\mathcal{T}}} \sum_{R\in\boldsymbol{\mathcal{R}}} P(s_t|T,s_{t-1},a_{t-1})P(R|T,h_t)P(T|h_t)\gamma^t R(s_t,\pi^*(h_t)) \\ &= \sum_t \sum_{s_t\in S} \sum_{T\in\boldsymbol{\mathcal{T}}} P(s_t|T,s_{t-1},a_{t-1})P(T|h_t)\gamma^t \left(\sum_{R\in\boldsymbol{\mathcal{R}}} P(R|T,h_t)R(s_t,\pi^*(h_t))\right).\end{aligned}$$

However, as noted in Equation 3.1, $P(R|T,h_t) = P_{b_{RT}}(R|T)$, since transitions observations give no information about rewards. Making the appropriate substitutions, we see that the term within parentheses is the expectation over π^*, conditioned on T, of $R(s_t,a_t)$:

$$\mathop{\mathbb{E}}_{h_t,T,R}\left[\sum_t \gamma^t R(s_t,\pi^*(h_t))\right] = \sum_t \sum_{s_t\in S} \sum_{T\in\boldsymbol{\mathcal{T}}} P(s_t|T,s_{t-1},a_{t-1})P(T|h_t)\gamma^t \mathbb{E}_{b_{RT}}\left[R(s_t,\pi^*(h_t))|T\right]. \tag{3.2}$$

Now, we define our new URTMDP, M' as follows. Let the state space, S', initial state distribution, s'_0, action set A', and transition set, $\boldsymbol{\mathcal{T}}'$, be the same as in M. Define the initial belief, $b_{R'T'}$, such that the marginal belief for each $T' \in \boldsymbol{\mathcal{T}}'$ has the following property:

$$P_{b_{R'T'}}(T') = P_{b_{RT}}(T') = \sum_{R\in\boldsymbol{\mathcal{R}}} b_{RT}(R,T).$$

In words, the marginal belief over each possible transition model is unchanged between M and M'. Let R^*_T be a reward function such that $R^*_T(s,a) = \mathbb{E}_{b_{RT}}\left[R(s,a)|T\right]$. Now, define the conditional belief over R' such that:

$$P(R'|T') = \mathbf{1}_{R'=R^*_{T'}},$$

where $\mathbf{1}_{R'=R^*_{T'}}$ is the indicator function for the event that R' is equal to $R^*_{T'}$. Now, define the new reward set $\boldsymbol{\mathcal{R}}'$ to be the set of all such R^* functions:

$$\boldsymbol{\mathcal{R}}' = \{R^*_T : T \in \boldsymbol{\mathcal{T}}\}.$$

With this construction, we have defined an URTMDP, M', whose initial belief over R, conditioned on T has support over only a single member of the reward set for all possible values of T.
Finally, we note that the expression for expected reward for a given policy on M' can be written as Equation 3.2. By assumption, π^* is the policy that maximizes this expression. Therefore, π^* is the optimal policy for M'.

$\square$

In order to find solutions to the URTMDP, we will reduce it to a POMDP, for which approximate solution techniques are known. In Theorem 3.1.4, we will prove that such a reduction exists for the class of URTMDPs described in the above theorem. [4] uses a similar reduction technique in the context of Bayesian reinforcement learning. We provide a version of the constructive proof here using the standard nomenclature for POMDPs, making the connection to the existing literature more apparent. This proof also differs from the one commonly used in the Bayesian RL literature in that it allows the agent to have a prior jointly over transitions and rewards. In contrast to the RL setting, the choice of reward function from $\boldsymbol{\mathcal{R}}$ is allowed to be dependent on the choice of transition model from $\boldsymbol{\mathcal{T}}$.

To start, we formalize what we mean when we say a URTMDP can be "reduced to" a POMDP.

Definition 3.1.3. *A URTMDP M can be reduced to a POMDP if there exists a POMDP M' such that the optimal policy for M', $\pi^*_{M'}$, is also the optimal policy for M.*

With that definition established, we move on to the proof.

Theorem 3.1.4. *For a URTMDP, M, if $P_{b_{RT}}(R|T)$ has support over only one element of $\boldsymbol{\mathcal{R}}$ for all $T \in \boldsymbol{\mathcal{T}}$ and $|\boldsymbol{\mathcal{T}}|$ is countable, then M can be reduced to a POMDP.*

Proof. Let M be a URTMDP where the conditional distribution $P_{b_{RT}}(R|T)$ has support over only one element of $\boldsymbol{\mathcal{R}}$ for all $T \in \boldsymbol{\mathcal{T}}$ and where $|\boldsymbol{\mathcal{T}}|$ is countable. We will construct a POMDP, M' with optimal policy π^* such that π^* is also the optimal solution for M.

We define the components of M' as follows:

- Let the action space of M' be the same as the action space of M: $A' = A$.

- Let the state space, $S' = S \times \boldsymbol{\mathcal{T}}$. Let σ_{ij} represent the ordered pair $\langle s_i, T_j \rangle$. These *pseudo-states* consist of a physical state, a state in the original state space, and information indicating which transition model that state follows. In a sense, our new state space consists of every possible instance of every state in S, where each instance has potentially different T.

- Define the reward function as $R'(\sigma_{ij}) = \sum_{R \in \boldsymbol{\mathcal{R}}} R(s_i) P(R|T_j)$. Since $P(R|T_j)$ has support over only one reward function, the reward for a state σ_{ij} is the value of the reward function in the support of $P(R|T_j)$ evaluated at s_i.

- Define the transition model as $T'(\sigma_{ij}, a, \sigma'_{kl}) = \mathbf{1}_{j=l} \cdot T_j(s_i, a, s_k)$, where $\mathbf{1}_{i=j}$ is the indicator function for $i = j$.[1] In words, our POMDP allows transitions only between pseudo-states corresponding to the same transition model, and the probabilities of these transitions are determined by the transition probabilities of the physical states under the corresponding transition model.

[1]Note that if $|\boldsymbol{\mathcal{T}}|$ is finite, then this transition model corresponds to a block-diagonal transition matrix. Each block corresponds to a particular transition matrix in $\boldsymbol{\mathcal{T}}$.

- Let the observation function be $O'(\sigma_{ij}) = s_i$. Since states are observed in the URTMDP, M, our POMDP agent can observe which physical state it is in, but it has to maintain a belief about which transition model applies in that state. This belief updates each time the agent observes a transition from one physical state to another.
- Let the initial belief distribution be $b_0(\sigma_{ij}) = \mathbf{1}_{s_i=s_0} \cdot \sum_{R \in \boldsymbol{\mathcal{R}}} b_{RT}(R, T_j)$.

We will now show that the expected reward under POMDP M' for a given policy, π, is equivalent to the expected reward of an agent using policy π in the URTDMP M. Note that the same π can operate on both M and M', since the observation history under M' is the fully-observed state history under M.

We start with the expression for the expected reward in M' for an agent following a fixed policy π:

$$\mathop{\mathbb{E}}_{h_t}\left[\sum_t \gamma^t R(s_t, \pi(h_t)) \middle| \pi\right] = \sum_t \sum_{\sigma_{ij} \in S'} \sum_{s \in S} P(\sigma_{ij}|T', h_t) P(s|\sigma_{ij}, O') \gamma^t R'(\sigma_{ij}, \pi(h_t)).$$

The left term here represents the agent's prior belief at time t that it is in pseudo-state σ_{ij}. The right term is a conditional probability of observing physical state s in pseudo-state $\sigma_i j$, given by the observation function O. Since $O(\sigma_i j)$ has support over only s_i, we can simplify the nested sum here:

$$\mathop{\mathbb{E}}_{h_t}\left[\sum_t \gamma^t R(s_t, \pi(h_t)) \middle| \pi\right] = \sum_t \sum_{\sigma_{ij} \in S'} P(\sigma_{ij}|T', h_t) \gamma^t R'(\sigma_{ij}, \pi(h_t)).$$

S' is the product space of S and $\boldsymbol{\mathcal{T}}$. We can rewrite the sum over S' as a dual sum over S and $\boldsymbol{\mathcal{T}}$. Further, we note that for a fixed transition model T_j, the reward function R' was chosen to be equivalent to the conditional expected reward function under the initial belief in M, b_{RT}.

$$\mathop{\mathbb{E}}_{h_t}\left[\sum_t \gamma^t R(s_t, \pi(h_t)) \middle| \pi\right] = \sum_t \sum_{T_j \in \boldsymbol{\mathcal{T}}} \sum_{s_i \in S} P(s_i|T_j, T', h_t) P(T_j|T', s_i, h_t) \gamma^t \mathbb{E}_{b_{RT}}\left[R(s_t, \pi(h_t))|T_j\right]$$

Because we assume the transition model is Markovian, the physical state, s_i, is independent of the history when conditioned on the previous physical state, the previous action, and T.

$$\mathop{\mathbb{E}}_{h_t}\left[\sum_t \gamma^t R(s_t, \pi(h_t)) \middle| \pi\right] = \sum_t \sum_{T_j \in \boldsymbol{\mathcal{T}}} \sum_{s_i \in S} P(s_i|T_j, s_{t-1}, a_{t-1}) P(T_j|T', h_t) \gamma^t \mathbb{E}_{b_{RT}}\left[R(s_t, \pi(h_t))|T_j\right] \tag{3.3}$$

Equation 3.3 tells us that the expected reward of a fixed policy in M' is the same as the expected reward given by the same policy in M, as shown in in Equation 3.2. We see, then, that the expected reward of an agent choosing actions according to a policy π is equivalent in both M and M', so that finding the optimal policy in the POMDP M' also gives the optimal policy for the URTMDP M.

□

Together, Theorems 3.1.2 and 3.1.4 allow us to reduce any URTMDP to a POMDP. In the next chapter, we will frame negotiable reinforcement learning as a URTMDP and use these reduction techniques to cast the problem as a POMDP. By reducing NRL to a decision problem, we admit planning-based solutions to the problem.

3.2 Generalized Uncertainty

In the previous section, we explored an MDP where the agent does not know T or R. Instead, the agent has a joint belief over transition and reward functions represented as a probability distribution over the product space of possible transition and reward functions. In a similar way, POMDPs assume that the environment state is not directly observable, and the agent has a belief represented by a distribution over the state space, S. In general, we can reason about decision problems where the agent has uncertainty about any combination of state, transition model, and reward function. As an example, consider the Uncertain Transition MDP.

Definition 3.2.1. *An Uncertain Transition MDP (UTMDP), M, is a URTMDP in which the agent's belief, conditioned on T, has support over only a single member of $\boldsymbol{\mathcal{R}}$. More precisely, for all $T \in \boldsymbol{\mathcal{T}}$, $P_{b_{RT}}(R|T) = \delta_{R^*}(R)$, where $R^* \in \boldsymbol{\mathcal{R}}$ and $\delta_{R^*}(R) = 1$ if $R = R^*$ and 0 otherwise.*

The UTMDP is well known, though not by the same name, in the Bayesian Reinforcement Learning literature [4], [8]. The UTMDP construction allows RL problems to be cast as decision problems, and the expected-reward-maximizing solution to this decision problem automatically trades off between exploration and exploitation in a Bayes-optimal way.

By enumerating the possible types of uncertainty, we get eight classes of U*MDP.[2] As we have seen, these classes are deeply related and can be reduced to one another. Some of these reductions are "free" in the sense that certain classes are a special case of others. Examples of this type of reduction include the reduction from an MDP to a POMDP with an observation function that deterministically reveals the true state to the agent and the reduction from UTMDP to URTMDP described in Definition 3.2.1. Other reductions require more work and result in problems of higher complexity. Examples include the reduction of a POMDP to an MDP, which results in a problem with a continuous state space and intractable transition model.

Figure 3.1 graphically explains all of the reduction relationships between these classes. Solid lines represent "free" reductions, and dotted lines represent reductions that are not implied by subset relationships. Several of the proofs required for the more difficult reductions are already included in this report or well known in literature, the rest are left as an exercise.

Figure 3.2 represents the same reduction relationships as Figure 3.1, but with edges that are implied by transitivity removed. It is easier to see a reduction path between any two U*MDP classes this way.

[2]By this naming convention, what we refer to commonly as a POMDP would be called a USMDP. This eliminates ambiguity as to the nature of the agent's uncertainty. We will continue to use the name POMDP for consistency with existing literature, but it should be noted that the two names are interchangeable.

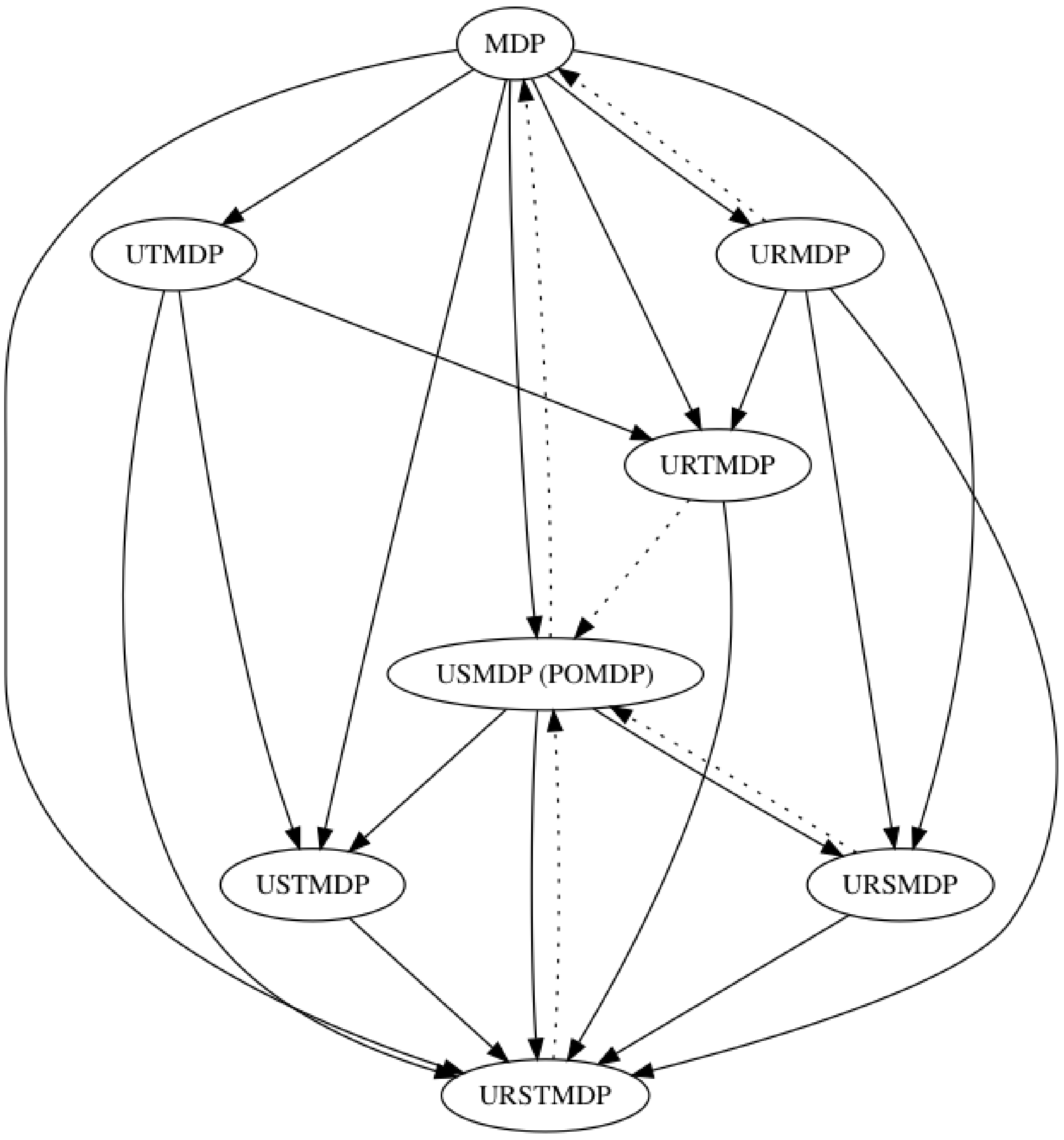


Figure 3.1: A graph showing the reduction relationships between U*MDP variants. Solid edges represent "free" reductions that result from subset relationships (e.g. MDP $\subset$ POMDP). Dotted edges represent possible reductions like the ones shown in Section 3.1.

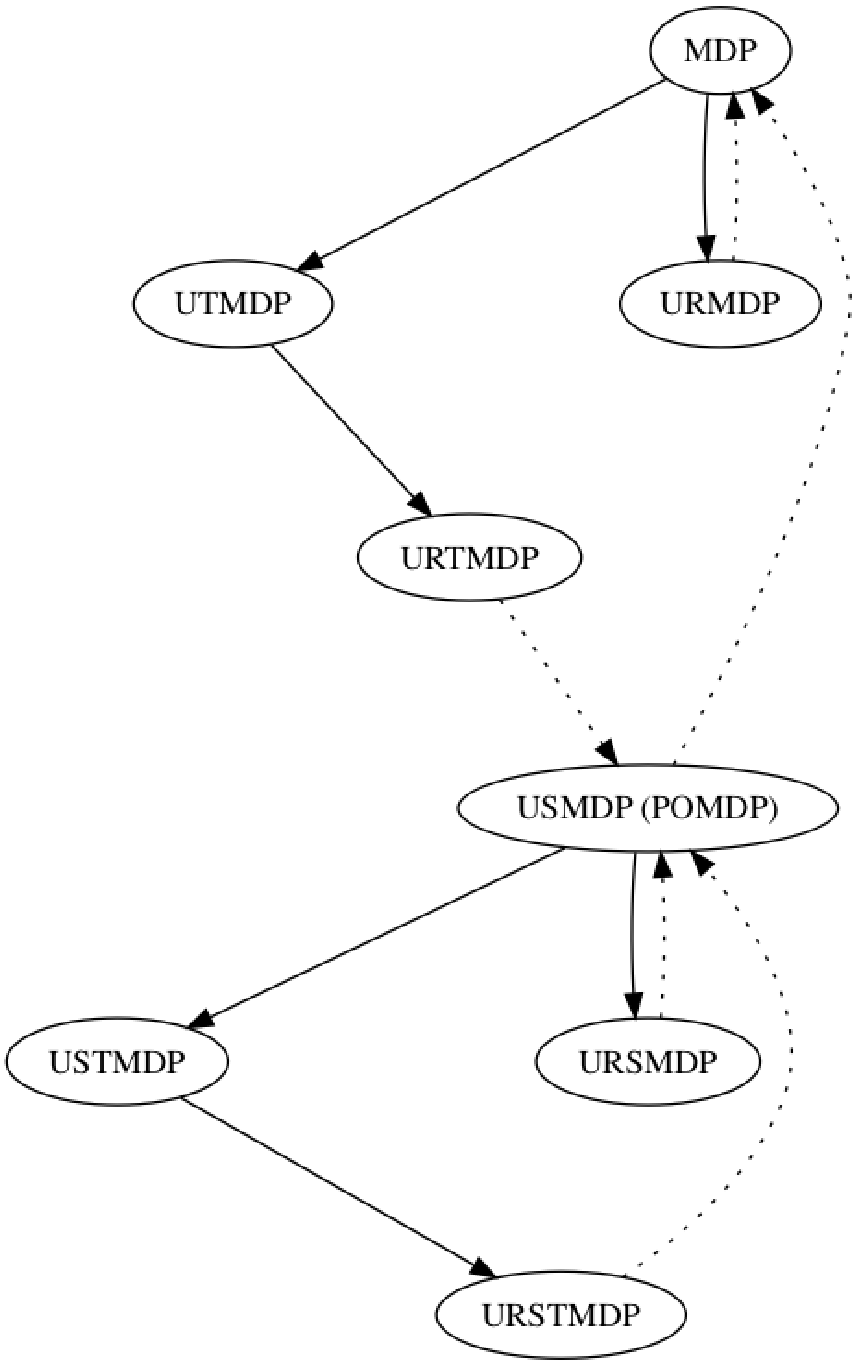


Figure 3.2: A simplified version of the graph in Figure 3.1, derived by removing edges already implied by transitivity relationships. The reduction path from any class to any other is apparent in this form.

Chapter 4

Applications and Evaluation

In this chapter, we apply the U*MDP framework developed in the previous chapter to negotiable reinforcement learning. First, we describe how a NRL problem can be viewed as a URSTMDP. Next, we discuss our experiment environment and framework. Finally, we discuss observations from the process of implementing a simple NRL agent.

4.1 Negotiable Reinforcement Learning as a URSTMDP

The NRL problem can be cast as a URSTMDP. Recall from Chapter 2 that in a NRL game, players parametrize their beliefs in terms of an observation function, and transition model. Each player also has their own utility function, which under stationarity assumptions can be rewritten as a reward function [15]. We will call these reward functions R_1 and R_2 for Players 1 and 2, respectively. Together, these elements can be seen as defining two POMDPs, one for each player.

From the perspective of the NRL agent, this problem can be recast as a URSTMDP where the agent has uncertainty about which POMDP it is acting in, and, consequently, which utility it should maximize. Concretely, we define a URSTMDP with the following elements:

- A and Ω are agreed upon by the players beforehand.
- $S = S_1 \cup S_2$: Since each player defines their own POMDP, we let our state space be the union of their two state spaces.
- $\boldsymbol{\mathcal{T}} = T_1, T_2$
- $\boldsymbol{\mathcal{R}} = R_1, R_2$
- $\mathcal{O} = O_1, O_2$: The set of possible observation functions is given by the corresponding functions of each POMDP.
- b_{RST}: The initial belief assigns probability w_1 to the reward-state-observation-function tuple $\langle T_1, R_1, O_1\rangle$, and with probability w_1 the initial state is drawn from s_0^1, the initial state distribution specified by Player 1. Likewise, with probability w_2, the reward-state-observation-function tuple is $\langle T_2, R_2, O_2\rangle$ and the initial state is drawn from s_0^2.

Because of the structure of the initial belief distribution, the URSTMDP agent either believes itself to be operating in Player 1's POMDP or in Player 2's POMDP. We can write the agent's belief state, then as a single probability p, which represents its subjective belief that it is operating in Domain 1. At any timestep, the posterior likelihood of Domain 1 is computed as

$$p_t = \frac{w_1 P_1(h_t)}{w_1 P_1(h_t) + w_2 P_2(h_t)}.$$

The expected reward of a policy starting from a history h_t and taking action a_t, then, is given by

$$\mathbb{E}[U|h_t, a_t, \pi] = \frac{w_1 P_1(h_t)}{w_1 P_1(h_t) + w_2 P_2(h_t)} \mathbb{E}[U_1|h_t, a_t, \pi] + \frac{w_2 P_2(h_t)}{w_1 P_1(h_t) + w_2 P_2(h_t)} \mathbb{E}[U_2|h_t, a_t, \pi].$$

This follows from the linearity of expectation. Linearity also allows us to factor out the denominators of these fractions. Since the optimal policy maximizes over this expression and the factored constant does not depend on the policy, we can ignore it. Thus, we see that an expected-utility-maximizing agent in this URSTMDP is a NRL agent.

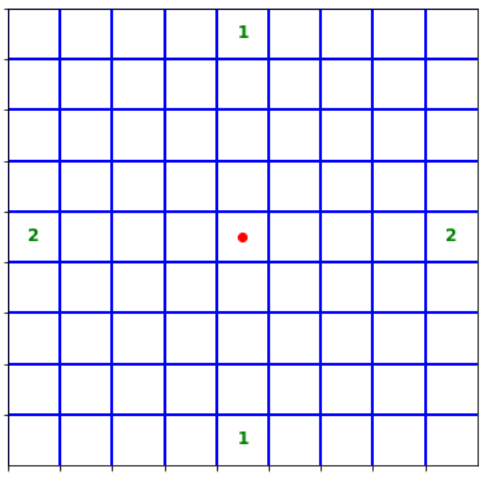
In the subsequent experiments, we will be using the URTMDP framework, ignoring state uncertainty for computational simplicity.

4.2 Experiment Environment

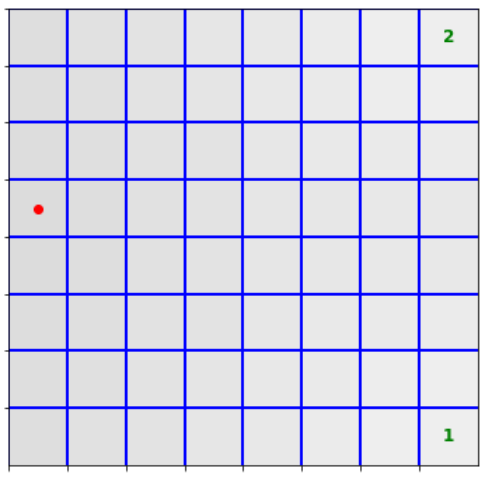
Our experiments are run in a modified version of the FrozenLake environment in OpenAI Gym [1]. FrozenLake is a grid world environment that simulates a goal MDP. The agent receives a reward upon reaching the specified goal position. The agent can choose to move `NORTH`, `SOUTH`, `EAST`, or `WEST`, and the transition model can be chosen to be either stochastic or deterministic. Under a stochastic transition model, simulating the eponymous frozen lake, the agent's action fails with probability 0.2, and the agent transitions into one of the unintended neighboring positions. In the FrozenLake environment, state is fully observed by the agent.

To experiment with a simple NRL agent, we make several modifications to this environment:

- We add a reward of -0.1 for each non-reward timestep.
- We add diagonal actions with a reward of $-(0.1 \cdot \sqrt{2})$. In the original environment, all trajectories of the same length incur the same cost, and this makes it difficult to identify when the NRL agent takes sub-optimal trajectories in the interest of compromise.
- We add multiple goals to the environment. In the original FrozenLake, there is a single goal state. In our version, we have two goal states, labelled `1` and `2`, corresponding to the utilities of Players 1 and 2. In some versions of the experiment, we have multiple goals for each player.



(a) Multi-Goal Environment



(b) Single Goal Environment

Figure 4.1: A visualization of the negotiable reinforcement learning gird world environment. The red dot represents the initial position of the agent. Player goals are labelled using respective player numbers. Left: A larger grid used to explore compromise behavior. Right: A smaller grid used to explore information gathering behavior. The smaller grid is used for computational simplicity.

- In the original FrozenLake, execution ends as soon as the agent reaches the goal. Because the goal is subjectively defined in the NRL setting, our execution keeps running until the agent takes an `EXIT` action.

All of our experiments take the same form. Player 1 assigns utility to the agent reaching each goal labelled `1` and has the belief that the environment has a deterministic transition model. Player 2 assigns utility to the agent reaching each goal labelled `2` and has the belief that the environment has a stochastic transition model. The agent is initialized with initial belief state w_1, corresponding to a subjective belief that the agent is in Player 1's MDP, M_1, with probability w_1 and Player 2's MDP, M_2, with probability $1 - w_1 = w_2$. This weight corresponds to the prior weight required by the NRL framework. We use point-based value iteration to learn a policy in the belief space. The agent is then placed in either M_1 or

M_2, and we observe as the agent acts according to its belief state, which is updated at each timestep. A visualization of the NRL environment is shown in Figure 4.1(b).

4.3 Experiments and Discussion

Observed Behavior In this set of experiments, we run the NRL agent in order to verify that its behavior resembles a type of bet-settling. In the first experiment, Player 1 believes that the environment is deterministic, and Player 2 believes the environment is stochastic. The true environment is chosen to be deterministic for this experiment. After running point-based value iteration with a belief set of 331 points, we execute the resulting policy in this environment. The agent's trajectory is seen in Figure 4.2. The purple arrows represent the agent's choice of action at each physical position under the current belief state. The color of each square represents the agent's subjective value at each position under the current belief. Because we are not using discounted rewards, all positions are close in value.[1]

Observing the agent's trajectory, we see that it initially moves towards goal `2`. However, each time an action succeeds, the agent's belief in the stochastic environment decreases. By the ninth frame, the agent's belief in the stochastic world, and as a result its belief that goal `2` grants reward, is low enough that the policy shifts to push the agent to goal `1`. This is the type of behavior we would expect of an agent that maximizes player utilities based on the likelihood of their beliefs.

Next, we use the same policy and place the agent in a stochastic world. The resulting trajectory is presented in Figure 4.3. The very first action results in a stochastic transition, and the likelihood of Player 1's belief immediately falls to 0.[2] At that point, the agent knows it is in Player 2's MDP and believes that goal `2` will give it reward. It heads to goal `2` accordingly.

[1]The policy under the initial belief is not optimal. Notice on the right side of the top row, two neighboring squares have opposing actions, resulting in an infinite loop if this policy were to be executed without updating the belief state. The inconsistent policy and inconsistencies in cell color representing the value of each position

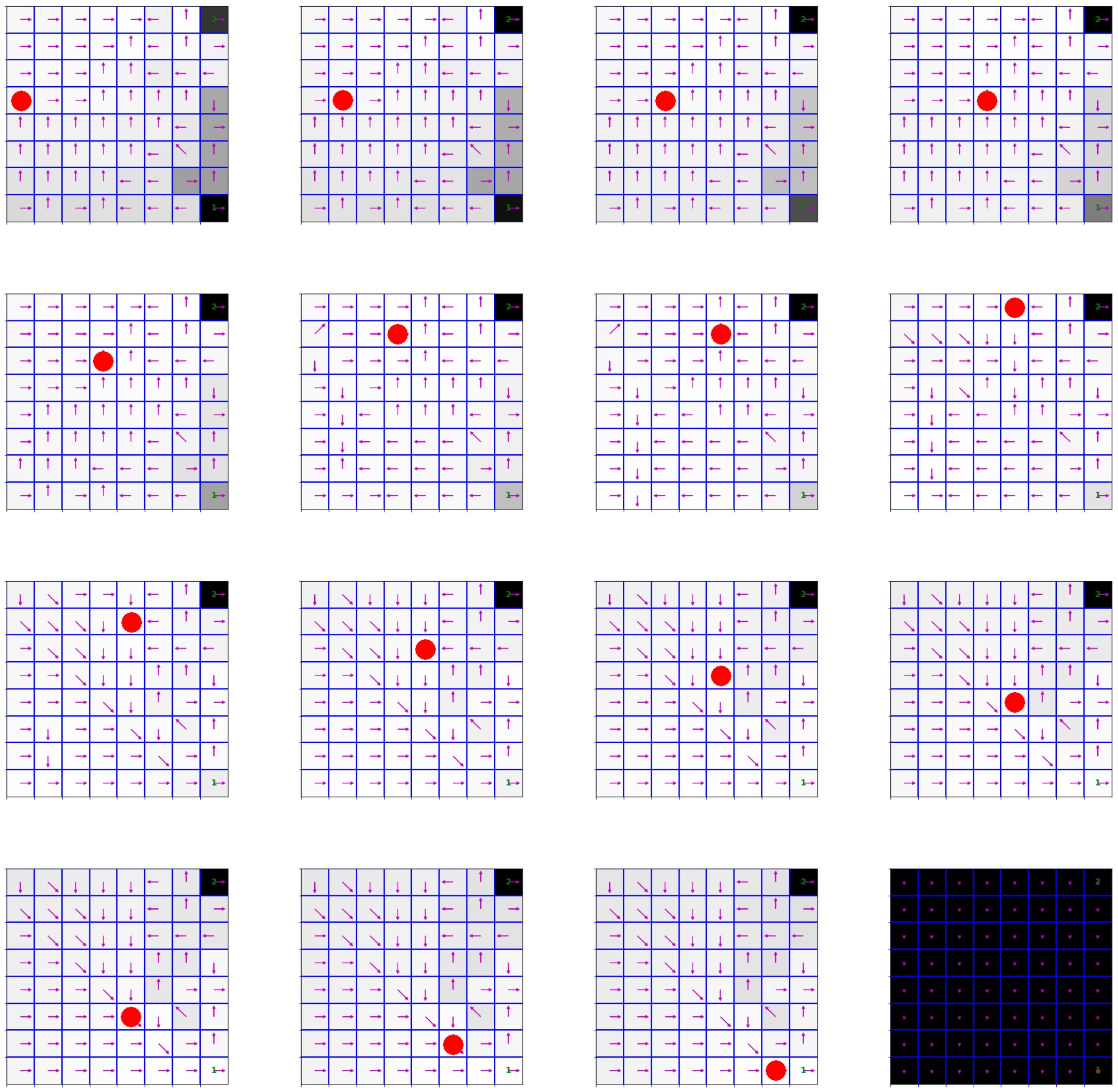


Figure 4.2: The agent initially heads for Goal 2 in the top-right corner. However, at Frame 9 it has observed eight deterministic transitions in a row. Its confidence in MDP 1 is high enough at that point that it veers downward and heads to Goal 1

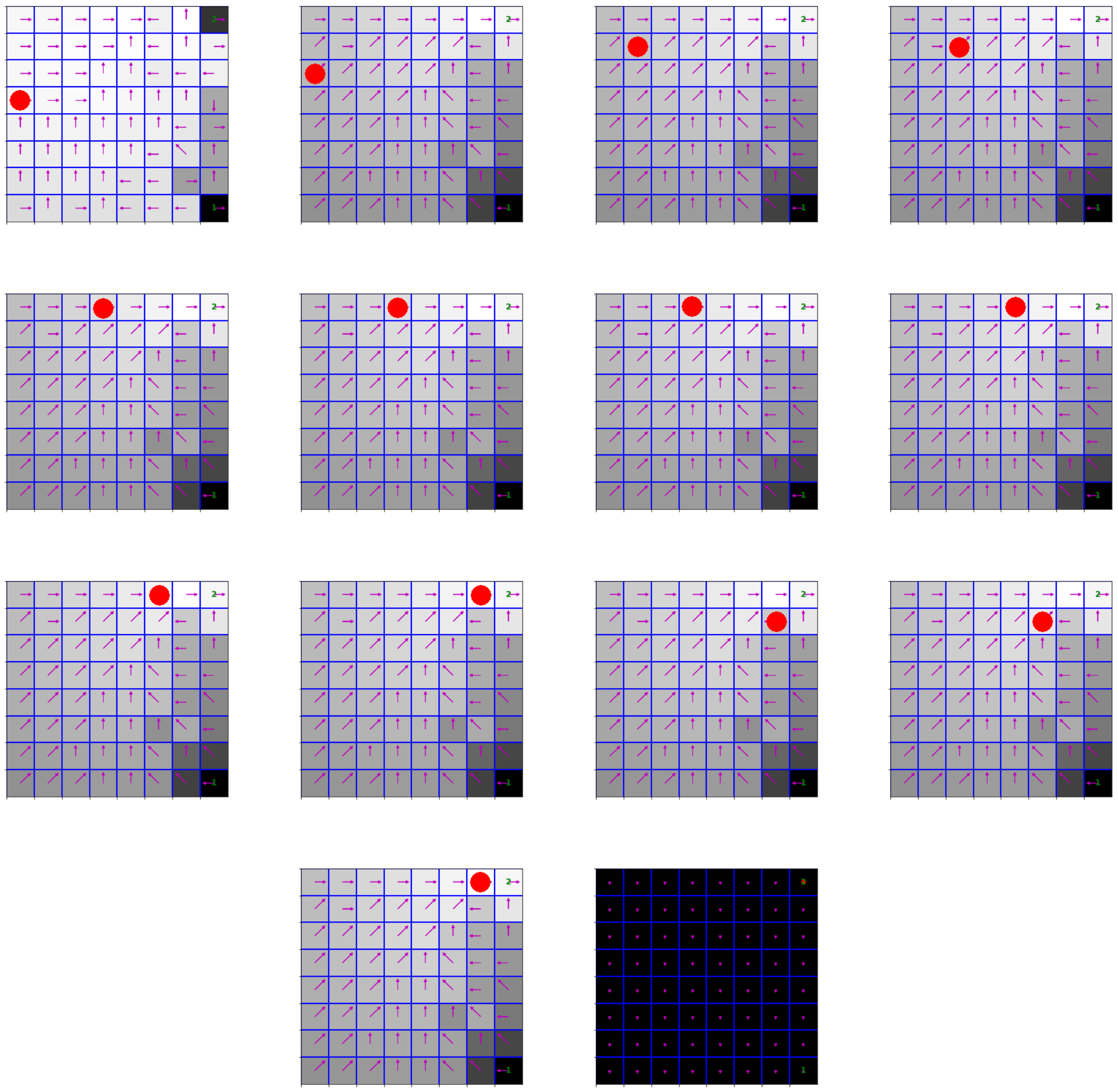


Figure 4.3: The agent immediately observes a stochastic transition, and its belief collapses onto MDP 2. It then moves directly to Goal 2, experiencing another stochastic transition at Frame 11.

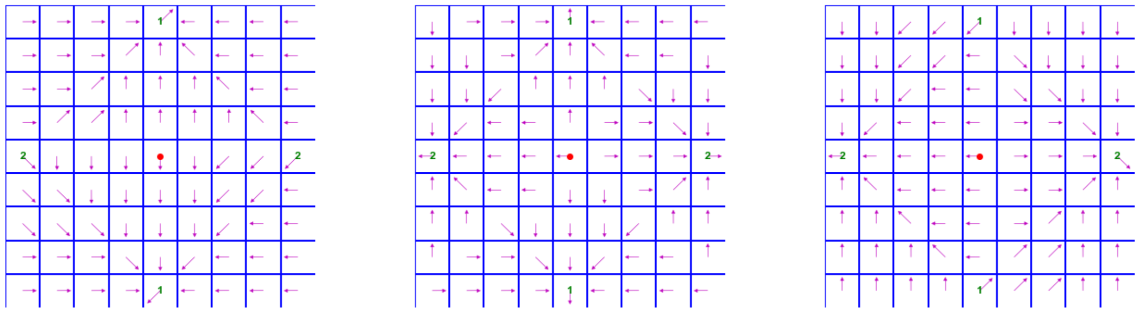


Figure 4.4: Left: A policy that only goes to goals labelled 1. Center: A policy that compromises by visiting all goals. Right: A policy that only goes to goals labelled 2.

Effect of Initial Weights In the NRL framework, prior weights w_1 and w_2 are treated as inputs to the planning algorithm and must be decided on by the players through negotiation. For this reason, it is valuable to understand how the NRL agent compromises between player goals as a function of this prior weight. For this set of experiments, we use an environment in which each player has two goal positions. The environment is visualized in figure 4.1(a). This environment is good for testing compromise behavior because reaching a goal for either player requires a significant deviation from the optimal path for reaching the other player's goals.

In order to isolate the effect of prior weights, we set both players' beliefs to be equal in this set of experiments. Recall that under equal weights, the NRL problem reduces to planning with a weighted sum of utility functions. We test for compromise behavior by observing the outcome of setting the prior weight for Player 1, w_1, to values between 0.0 and 1.0, going in increments of 0.05. For low values of w_1, we should expect to see a policy that favors goals labelled 2. For high values, we expect the policy to choose goals labelled 1. For intermediate values, the agent should choose to visit all goals, since it is uncertain which MDP it is acting in. Examples of each of these policies are visualized in Figure 4.4.

are artifacts of the point-based approximation.

[2]A note to future NRL players: try not to assign probability 0 to events that actually happen.

Figure 4.5 shows the number of each player's goals reached by the agent at various settings of w_1. We see that the agent prefers compromise behavior for a wide range of values. For $w_1 \in [0.1, 0.9]$, the agent chooses to visit all four goals. Only for w_1 very close to 0.0 or 1.0 does the agent only visit the goals of one player. At w_1 close to 0.05, the agent visits both of 2 goals and one of the 1 goals with analogous behavior for w_1 close to 0.95. We see that the NRL agent chooses compromise over commitment to a single set of goals for the majority of prior weight values.

It is important to consider the results of the previous experiment in interpreting this outcome. The experiment summarized in Figure 4.5 assumes agents' beliefs are *identical*. In the case that agents have different beliefs, the NRL agent may over time strongly optimize one objective and neglect the other. In particular, when one player makes significantly more accurate predictions than the other, the NRL agent will not compromise as much as it does in the above experiment, preferring instead to optimize the better predictor's utility. In the most extreme case, if a player assigns probability 0 to even a single observed state transition, the NRL agent will cease to consider that player's utility in its search. This effect is seen in Figure 4.3. The net effect of this property is that any parties wishing to build a NRL agent will have to expend significant resources on building accurate predictions, creating a predictive arms race of sorts.

Subjective State Value During Execution The promise of compromise behavior may bring parties into agreement over the decision to build a NRL agent, but if during execution parties feels that the agent will not take actions that guarantee them high future rewards, those parties may be tempted to end cooperation. The next experiment attempts to test how losing parties assess the agent's behavior during execution.

To answer this question, we turn back to the trajectories presented in Figures 4.2 and 4.3. In the first trajectory the agent is, in fact, operating in Player 1's MDP. We ask at each timestep what Player 2 believes their expected sum of future rewards to be. Recall that each player believes the agent is acting in their own MDP. Since both players know the agent's

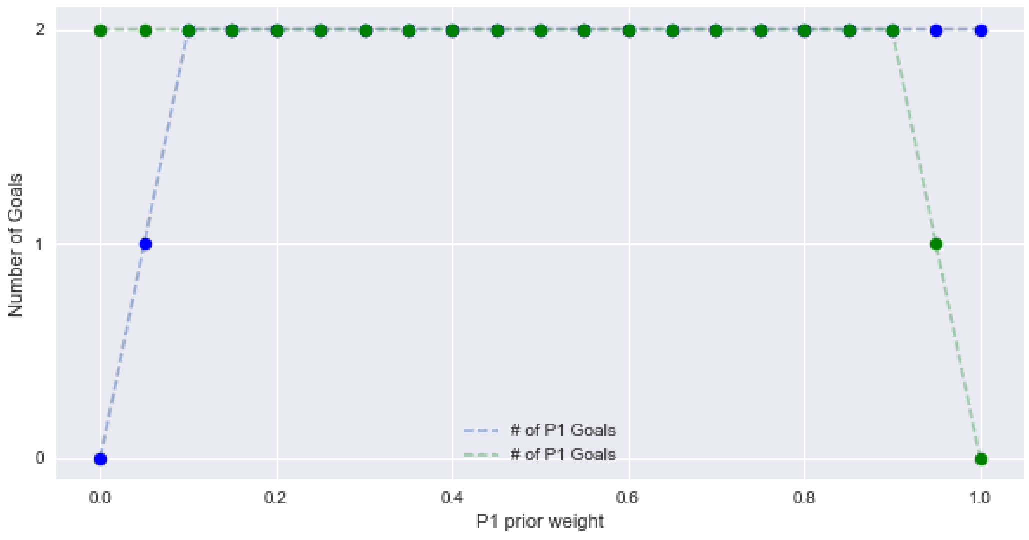
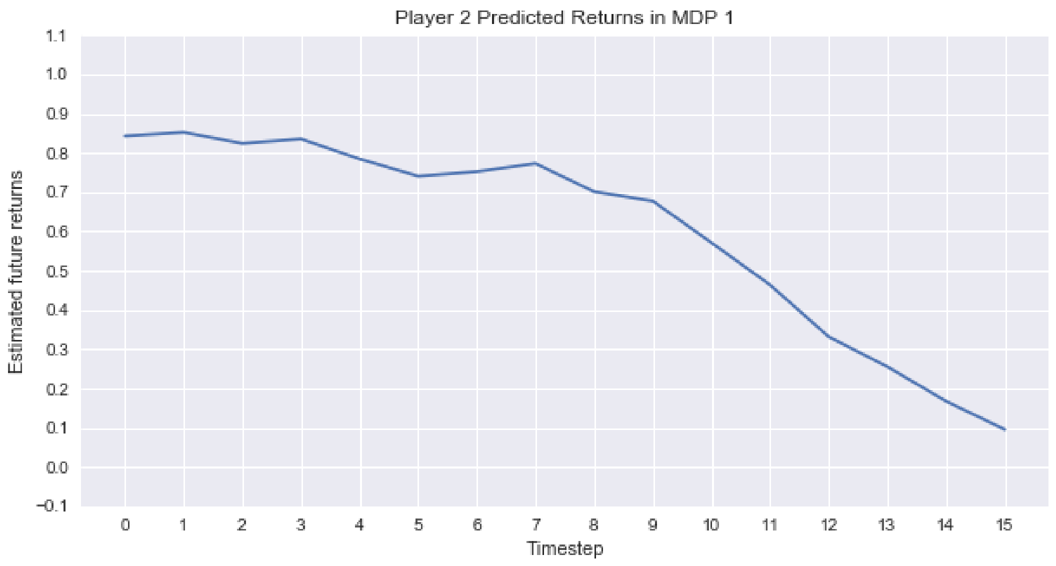


Figure 4.5: Number of player goals reached as a function of initial weight w_1.

policy, physical state, and belief state at each timestep, each player can estimate their future rewards by simulating the agent acting in their respective MDP with initial configuration given by the current configuration. Concretely, consider the trajectory of Figure 4.2. At timestep 0, the agent is in physical position 24 with belief state $w_1 = 0.5$ and a policy learned using PBVI. Player 2 can estimate their expected reward by placing the agent in position 24 with belief $w_1 = 0.5$ within Player 2's own MDP and simulating the agent's behavior.

For the experiments here, we simulate the agent 100 times in each configuration encountered along the two trajectories. For each simulation, we place the agent within the losing player's MDP and measure the average reward attained by the agent from that starting configuration. Player 2's assessment of their expected future rewards during execution in MDP 1 is shown in the top frame of Figure 4.6. Player 1's assessment of their expected future rewards during execution in MDP 2 is shown in the bottom frame.

We see that until frame 9, Player 2 believes that the agent has a high chance of reaching goal 2. It is only when the agent's behavior shifts and if veers towards goal 1 that Player



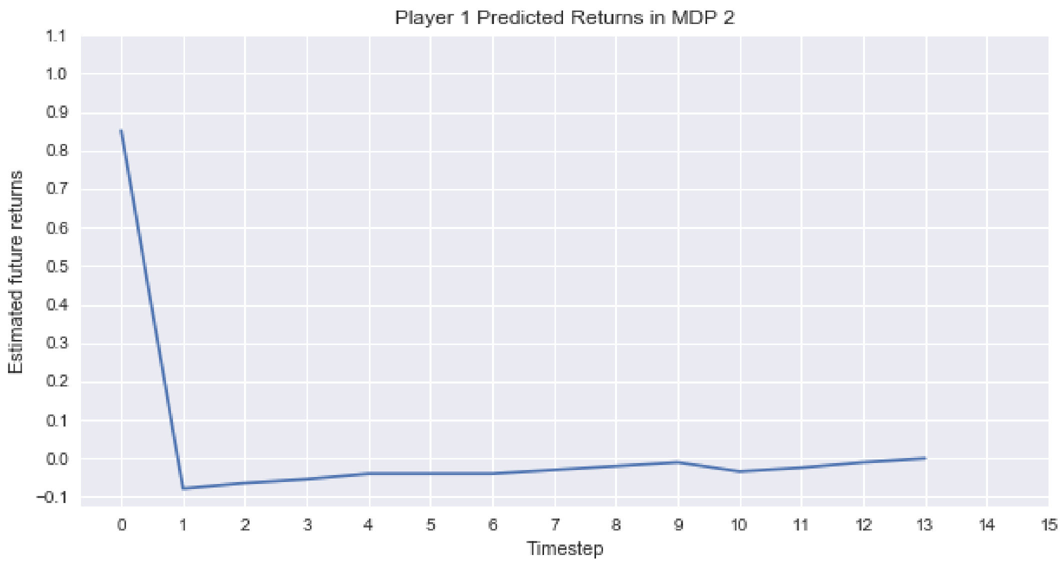


Figure 4.6: Top: Player 2's predicted expected reward during trajectory depicted in Figure 4.2. Bottom: Player 1's predicted expected reward during trajectory depicted in Figure 4.3.

2 begins to assess their chances of receiving rewards as decreasing sharply. In contrast, as soon as Player 1 observes that $w_1 = 0.0$, they realize that the agent will never reach goal `1`. Both of these assessments are consistent with what we might predict. In Player 2's case, we would not expect that the agent will in fact go to goal `1` until it makes a sharp turn away from goal `2`, and this does not happen until the agent is highly confident that it is in MDP 1. In Player 1's case, we know as soon as the agent assigns a probability of 0.0 to MDP 1 that it will never go to goal `1`.

Chapter 5

Related Work

Because of the generality of the U*MDP framework, many specialized models for planning under uncertainty can be reduced to U*MDP problems. In this chapter, we discuss two groups of models that are particularly relevant to the present work. In the first section, we discuss Bayesian Reinforcement Learning, which uses reductions similar to those in Chapter 3 in order to map problems with uncertain dynamics into problems with state uncertainty, where solution techniques are well-understood. In the second section, we discuss Multi-Objective Reinforcement Learning, which is sometimes concerned with learning Pareto optimal policies for solving sequential decision problems while balancing multiple objectives. This goal is similar to the NRL problem we have previously explored.

5.1 Bayesian Reinforcement Learning

Reinforcement learning looks at the problem of learning behaviors that attain high rewards in MDPs often under the assumption of unknown dynamics [21]. Some Bayesian approaches to reinforcement learning can be modelled withing the U*MDP framework. A subset of Bayesian reinforcement learning methods treats the unknown dynamics as an unobserved random variable and keeps track of explicit beliefs about the MDP transition function [8].

The earliest versions of this approach to RL come from control theory in an approach known as adaptive dual control methods [5], [6]. The name refers to the fact that controllers for these problems have to serve the dual purposes of exploration and reward maximization, which are not always complementary behaviors. These approaches fit in the U*MDP framework as UTMDPs, and a simplified version of the reductions shown in Chapter 3 is often used to reduce these problems to POMDPs [8]. It has been shown [4] that using offline planning methods for solving these reinforcement learning problems results in Bayes optimal exploration-exploitation trade-offs.

These models can also be used to build risk-aware agents for safer exploration in RL settings [11]. Rather than attempting to maximize expected returns, these agents can choose actions to actively minimize the risk they are exposed to. The risk minimization framework can also be described as a UTMDP, and risk-minimizing U*MDP agents are a compelling future line of research.

5.2 Multi-Objective Reinforcement Learning

Multi-Objective Reinforcement Learning (MORL) describes sequential decision problems where an agent's goal is to behave optimally with respect to a vector-valued reward signal $R = \langle R_1, \ldots, R_k \rangle$[7]. Methods for solving these problems split into two broad categories: the first finds optimal policies by imposing an ordering on the space of vector-valued rewards, the second chooses a vector of weights $\langle w_1, \ldots, w_k \rangle$ and treats the weighted sum of reward components as a single reward function to be optimized [13]. Of these, the latter is most closely related to the work presented here. If these weights are chosen to be non-negative and sum to 1, then the resulting policy is also Pareto optimal [18]. The particular choice of weights is exogenous to the problem of finding a policy. This approach to MORL corresponds exactly to Harsanyi's approach in the economics literature and also to NRL in the case where both players have the same beliefs [3], [9]. Though not explicitly Bayesian, these methods bear an algebraic resemblance to the URMDP formulation. As a result, these

MORL techniques can mathematically be reduced to solving a U*MDP problem as presented in this work.

Chapter 6

Conclusion

Negotiable reinforcement learning has the potential to be an important tool for the future cooperative use of artificial agents. However, the algorithmic framework does not yet exist to connect NRL to existing methods in AI. Beginning to bridge that gap is the focus of the present work. The contributions of this report are summarized below.

We've presented the U*MDP framework for modelling uncertainty about all components of a Markov decision process. We then showed that U*MDPs can be solved using algorithms for planning under state uncertainty. In doing so, we've reduced a wide class of problems in the AI literature related to solving decision problems with partial knowledge to the problem of solving a POMDP.

Of particular interest to us is the problem of negotiable reinforcement learning. This framework gives us a technique for balancing the utilities of multiple stakeholders by optimizing their respective utilities in proportion to the likelihood of their predictions. The URTMDP model allows us to tie this framework back into the existing AI literature and use existing techniques to implement a simplified NRL agent.

Upon implementing such an agent, we make several important observations. First, we note that the NRL agent behaves as we would expect: it serves the goals of the player that makes correct predictions about the agent's observations. We also observe that when unable to make a highly confident prediction about the correct reward function, the NRL agent chooses compromise for a wide range of confidence levels. Furthermore, when the agent's behavior is viewed through the lens of the players, we see that as long as a player's predictions are consistent with the agent's observations, the player is likely to remain satisfied with the agent's actions during execution.

The NRL agent implemented here is simplified for easy analysis. Moving forward, attempting more sophisticated NRL agents will require work on three fronts. First, we will need to apply faster and more accurate POMDP algorithms to the NRL setting. Simulation-based approaches to POMDP planning are promising [16]. Second, a strong algorithm for preference elicitation is necessary. Recent work on this front shows encouraging results [2]. Finally, work needs to be done on building practical systems for *belief* elicitation.

References

[1] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, *OpenAI Gym*, 2016. eprint: `arXiv:1606.01540`.

[2] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” *ArXiv e-prints*, Jun. 2017.

[3] A. Critch, “Toward negotiable reinforcement learning: shifting priorities in Pareto optimal sequential decision-making,” *ArXiv e-prints*, Jan. 2017.

[4] M. O. Duff, “Optimal Learning: Computational Procedures for Bayes-adaptive Markov Decision Processes,” PhD thesis, 2002.

[5] A. Feldbaum, “Dual Control Theory, Parts I and II.,” *Automation and Remote Control*, no. 21, 1961.

[6] N. Filatov and H. Unbehauen, “Survey of Adaptive Dual Control Methods,” *IEE Proc.-Control Theory Appl.*, vol. 147, no. 1, pp. 118–128, Jan. 2000.

[7] Z. Gabor, Z. Kalmar, and C. Szepesvari, “Multi-Criteria Reinforcement Learning,” in *International Conference on Machine Learning (ICML-98)*, 1998.

[8] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar, “Bayesian Reinforcement Learning: A Survey,” *ArXiv e-prints*, Sep. 2016.

[9] J. C. Harsanyi, “Cardinal Welfare, Individualistic Ethics, and Interpersonal Comparisons of Utility,” *Journal of Political Economy*, vol. 63, no. 4, pp. 309–321, 1955.

[10] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and Acting in Partially Observable Stochastic Domains," *Artif. Intell.*, vol. 101, no. 1-2, pp. 99–134, May 1998.

[11] S. Mannor, D. Simester, P. Sun, and J. N. Tsitsiklis, "Bias and Variance Approximation in Value Function Estimates," *Management Science*, vol. 53, no. 2, pp. 308–322, 2007.

[12] J. Pineau, G. Gordon, and S. Thrun, "Point-based Value Iteration: An Anytime Algorithm for POMDPs," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, ser. IJCAI'03, Acapulco, Mexico, 2003, pp. 1025–1030.

[13] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A Survey of Multi-Objective Sequential Decision-Making," *ArXiv e-prints*, Feb. 2014.

[14] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online Planning Algorithms for POMDPs," *J. Artif. Int. Res.*, vol. 32, no. 1, pp. 663–704, Jul. 2008.

[15] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. 2010.

[16] D. Silver and J. Veness, "Monte-Carlo Planning in Large POMDPs," in *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, 2010, pp. 2164–2172.

[17] T. Smith and R. Simmons, "Heuristic Search Value Iteration for POMDPs," in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, ser. UAI '04, 2004, pp. 520–527.

[18] H. Soh and Y. Demiris, "Evolving Policies for Multi-reward Partially Observable Markov Decision Processes (MR-POMDPs)," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, 2011, pp. 713–720.

[19] E. J. Sondik, "The Optimal Control of Partially Observable Markov Processes over the Infinite Horizon: Discounted Costs," *Oper. Res.*, vol. 26, no. 2, pp. 282–304, Apr. 1978.

[20] M. T. J. Spaan and N. A. Vlassis, "Perseus: Randomized Point-based Value Iteration for POMDPs," *J. Artif. Intell. Res.*, vol. 24, pp. 195–220, 2005.

[21] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning.* 1998.