Policy teaching through reward function learning

A thesis presented

by

Haoqi Zhang

to

Computer Science and Economics

in partial fulfillment of the honors requirements
for the degree of
Bachelor of Arts
Harvard University
Cambridge, Massachusetts

April 3rd, 2007

Policy teaching through reward function learning

Abstract

Many situations arise in which an interested party wishes to provide incentives to an agent so that the agent's modified behavior meets some goal external (but possibly related) to the agent. The problem is difficult because the reward function governing the agent's behavior is often complex and unknown to the interested party. We study this problem in sequential decision-making tasks modeled by Markov Decision Processes. We develop novel algorithms that learn the reward function of an agent and use learned information to provide incentives for the agent to meet an external goal. We conduct experiments motivated by a real world scenario, and show that we can generally find incentives to modify an agent's behavior as desired in a small number of iterations. In addition to studying a technical problem, this work has applications in education, economics, and multi-agent systems.

Acknowledgments

I thank David Parkes for being a great mentor. His advice, direction, and support have not only made this senior thesis possible, but have helped me grow as a researcher, a teacher, and a person. The weekly meetings helped, a lot. I thank Jerry Green for his support and guidance, and for helping me realize the significance of this work in an economics context. I thank Daniel Wood and Jonathan Kolstad for reading drafts of this work and providing useful feedback. I thank Alex Allain for taking an interest in this work and for pushing me to think about the work in realistic settings. I thank Andrei Munteanu for helping me visualize space. I thank Pavithra Harsha for her help on programming formulations.

Lastly, I thank Laura and my parents.

All errors are my own.

Contents

Chapter 1

Introduction

There are many scenarios in which an interested party wishes for an agent to behave in a certain way. An elementary school teacher wants his student to solve arithmetic problems correctly. Parents want their child to go home after school. A firm wants a consumer to make a purchase. In many cases, the behavior desired by the interested party may differ from the actual behavior of the agent. The student may solve the problems incorrectly, the child may go to the park instead of going home, and the consumer may not be interested in the product being sold.

For the interested party to affect the behavior of an agent, the interested party can often provide incentives to modify the agent's preferences so as to induce a behavior that meets the external goal. A teacher can offer candy to a student for solving problems correctly, parents can allow more TV time if their child comes home immediately, and a firm can use advertisements and discounts to entice the consumer to make a purchase.

However, there are a number of underlying obstacles that make it difficult for the interested party to modify the agent's behavior as desired. Some of the major issues include:

- **The amount of incentive that can be given may be limited.**
  A teacher may only have so much candy; parents may only wish to give so much TV time; a firm may only wish to spend so much on advertisements

and discounts.

- **Providing incentives after completion of tasks may not be sufficient.**
  An agent will be most interested in immediate rewards, and offering incentives later will be discounted by the agent. A student may value finishing the problems quickly over receiving candy after the homework gets graded; a child may prefer playing at the park now to watching TV later; a consumer may prefer immediate discounts over mail-in rebates.

- **The agent's preferences may be complex.**
  The agent's behavior is a sequence of actions in a complex domain, where a decision now affects subsequent decisions. Figuring out where and when to offer incentives can be difficult, especially when getting the agent to perform as desired may require additional incentives in many states. In some situations, the agent's actions may be unobservable in certain states.

- **The agent's preferences may be unknown to the interested party.**
  One cause of this may be that the agent wishes not to share his preferences with the interested party. Another cause is that it may be difficult for the agent to write down his preferences. The student would be hard-pressed to write down relative weights for different alternatives; the child cannot write down how much he values going to the park, and a consumer cannot easily write down how much something is worth to him. While the agents' actions are governed by their inherent preferences, writing down such information accurately is extremely difficult, especially in complex domains.

Despite these obstacles, there is considerable interest in understanding how to provide incentives to affect the behavior of agents. This thesis aims to answer the question of **policy teaching**: Given an agent facing a sequential decision-making task, how can an interested party provide minimal incentives to modify the agent's behavior to meet a goal external to the agent?

In addition to the numerous real-world applications, this work is motivated by applications to multi-agent systems, where methods for learning the participants' unknown preferences and providing incentives to modify the agents' behaviors can aid in the design of better systems and implementation of socially beneficial outcomes.

1.1 Contributions

We study the policy teaching problem under the Markov Decision Process (MDP) framework. We tackle the policy teaching problem with known and unknown rewards, both in domains with an expert and domains without. For policy teaching with unknown rewards, we use techniques of inverse reinforcement learning [16] as a starting point to develop a novel method for elicitating the preferences of an agent using only observations of the agent's policy before and after added incentives. Based on this method, we develop algorithms that iteratively learn the reward function of an agent, and use learned information to provide incentives to meet an external goal. We prove bounds on the number of elicitation rounds our algorithms require before returning a solution to the policy teaching problem if one exists.

The algorithms we have developed are general elicitation methods that can allow for any elicitation strategy. We discuss possible objective functions for the elicitation process, and introduce tractable heuristics that can significantly decrease the number of elicitation rounds necessary in practice. We perform experiments on our algorithms with these heuristics in a simulated domain inspired by a real-world example. Our results show that even with unknown rewards, our algorithms can generally find incentives to modify an agent's behavior as desired within a small number of iterations.

Finally, we provide a sketch of applications of policy teaching to real-world

problems and to the study of multi-agent systems. We address the major critiques of our work, and discuss extensions and open questions for future research.

1.2 Related Work

Our work is interdisciplinary in nature, and related to research in economics and computer science. The closest analogue is in economics, where contract theory and principal-agent theory study the question of how a principal can provide incentives in the form of contracts to align the interest of the agent with that of the principal. Principal-agent problems often deal with hidden information (adverse selection) and hidden action (moral hazard), and contract theory deals with this by using incentive compatibility constraints such that the agent will pick the contract that is optimal for the principal and then act in a desired way. Bolton and Dewatripont [4] provide a thorough coverage of this area of study, as does Laffront and Martimort [13].

The questions addressed by principal-agent theory is similar to the policy teaching question. As in principal-agent theory, we consider self-interested agents who perform with respect to their own preferences and not the preferences of the interested party. In our work, we can view the interested party as a principal who provides incentives (a contract) to drive the agent toward behaving as desired. Furthermore, in considering domains with unknown preferences, we study an adverse selection problem and attempt to come up with an optimal contract despite the hidden information. Like dynamic adverse selection in contract theory, we consider settings with repeated interactions between the interested party and the agent.

Despite these similarities, there are a number of fundamental differences in the domains considered and assumptions made. First, an agent's preferences in sequential decision tasks are highly multidimensional; generalizing to multidimen-

sional domains has been difficult in contract theory. Second, we assume that the states and actions of the agent are observable, and thus do not study moral hazard problems. Third, we consider a repeated setting in which the agent is myopically rational; the agent will not attempt to distort the interested party's beliefs about his preferences by purposely performing suboptimally. This is different than dynamic adverse selection, where the agent is strategic and may act to deliberately withhold information from the principal. Finally, the use of incentive constraints in domains with hidden information in contract theory generally results in less efficient allocations than under complete information. Our work attempts to learn the hidden preferences of the agent to enable us to provide better contracts.

In computer science, our work is inspired by applications of inverse reinforcement learning to apprenticeship learning. Abbeel and Ng [1] studied this problem by extracting a reward function from a human expert, and using the reward function to govern the behavior of a machine agent. They conducted experiments in a driving setting, where their machine agent avoided obstacles while driving by performing based on a reward function extracted from demonstrations by a human expert. By learning a reward function instead of just mimicking policies, their agent was able to perform well even when traffic conditions were modified.

Part of our work can be seen as an extension to AI apprenticeship learning, in that we aim to teach a human agent to behave like the human expert. In doing so, our work faces numerous challenges that were not present in Abbeel and Ng's work. First, in teaching a human agent, we cannot redefine the agent's reward function at will; we may only provide incentives to induce the agent to behave according to both his inherent reward function and the provided incentives. Second, given the limits on the amount of incentives that can be provided, it may not be possible for the agent to perform like the expert. Finding the best alternative can be computationally difficult. Finally, because we must provide incentives, the size of the incentives must be in line with the reward function of the agent. For

example, if the agent is buying a car, providing a five dollar discount would not be sufficient, but we may nevertheless make this mistake if we do not learn both the shape and size of the agent's reward. In Abbeel and Ng's work, any reward function within the solution space was sufficient, out of which they picked the one that would generalize well.

As our work is concerned with learning the preferences of the agent, it is related to the literature on preference elicitation. The literature is vast and we will not review it all here, but it is worth pointing out some similarities and differences in comparison with our work. Often, preference elicitation is performed by asking a series of queries [7, 8, 17] about the agent's preferences, based on which the elicitor gathers information to place bounds on the utility function of the agent. In our work, we do not query for the agent's preferences directly, but instead perform *indirect elicitation* by using observations of the agent's responses to incentives and the principle of revealed preference to place bounds on the agent's utility functions[1]. While indirect elicitation techniques based on the principles of revealed preferences are nothing new (in fact, techniques in inverse reinforcement learning is an example), such techniques are typically passive; they are applied to observed behaviors, and are unconcerned with generating new evidence based on which to make further inferences about the agent's preferences. Our elicitation method offers incentives to an agent, and actively generates new evidence about the agent's reward function based on his response to the provided incentives. To our knowledge, this approach has not been previously studied in the literature, and may be useful for learning preferences in a wide range of settings.

Preference elicitation is commonly seen as a costly process, and the literature often adapts criteria to balance this cost with the benefits of the elicited information. In our work, we show that our elicitation algorithms can adapt the commonly used minimax regret decision criteria [25] as an objective function for the elicitation

[1]For a survey of the economics literature on revealed preferences, see Varian's article [24].

process. However, computing this objective function is intractable in our domain, leading us to focus primarily on developing tractable heuristics that aim to reduce the length of the elicitation process in practice.

Other areas of computer science have also made use of preference elicitation techniques. For example, work by Gajos and Weld [12] on personalizable user interfaces used preference elicitation to learn an accurate cost function based on which to generate user interfaces. More recently, Gajos, Long, and Weld [11] extended their work to generate custom user interfaces for users with disabilities. Their work differs from ours in that their elicitation method queries the agent's preferences directly, and in that they are not concerned with sequential decision tasks.

The policy teaching problem bears resemblance to imitation learning [18, 19, 23], which aims to aid the transfer of knowledge from an expert to a student in reinforcement learning. Our work differs in that we assume the agent is a planner who already knows how to perform optimally with respect to his preferences, but that the interested party may still wish to provide incentives to affect the agent's optimal behavior. Furthermore, a student in imitation learning needs to reason about how similar the expert is to him; in our setting, the student cares only about his own behavior and does not reason about the expert's behavior.

Finally, our work is inspired by the numerous applications of computer science and economics techniques to real-world problems like those which motivate our work. There is work applying artificial intelligence techniques to education [2]. There is work that uses a Markov Decision Process to design a computerized guidance system that determines when and how to provide prompts to a user with dementia [3]. Works in these areas have been successful, and we believe could benefit from better understanding of the agent's preferences.

1.3 Outline

In Chapter 2, we introduce Markov Decision Processes and study the policy teaching problem with observable rewards. In Chapter 3, we introduce Inverse Reinforcement Learning and study the policy teaching problem with unknown rewards. In Chapter 4, we present the results of our experiments. In Chapter 5, we discuss potential applications, critiques, extensions, and open questions for future research.

Chapter 2

Policy Teaching with Known Rewards

In this chapter we introduce Markov Decision Process (MDP) as a framework for modeling sequential decision tasks, and use this framework to discuss the policy teaching problem with known rewards.

2.1 Markov Decision Process

As we have shown in our introduction, many of the examples we have considered are sequential decision-making tasks, where decisions made now can affect decisions in the future. In modeling such tasks, we are interested in a framework that can capture the following:

- **Different states of the world**.
  We want to know if the child at school, at the park, or at home.
- **Rewards for different states**.
  How much does the child value being at the park? Being at home?
- **Actions moving agents from state to state**.
  Walking east brings the child from one place to another.
- **Uncertainty in an action's outcome**.
  The child may attempt to walk east but end up walking south.

- **Discounting**.

  Future rewards count less than immediate rewards.

- **Variable time period**.

  We are interested in what the child does, and do not want to be limited to only considering what the child does in a fixed number of steps.

The *infinite-horizon Markov Decision Process* framework fulfills our criteria quite nicely. Not only does the framework allow us to model complex domains, it also has an elegant, intuitive formalism.

2.1.1 Definition

Definition 1. An *infinite horizon MDP* is a model $M = \{S, A, R, P, \gamma\}$:

- S is the set of states in the world.
- A is the set of possible actions.
- R is a function from S to $\mathbb{R}$, where $R(s)$ is the reward in state s.
- P is a function from $S \times A \times S$ to $[0, 1]$, where $P(s, a, s')$ is the probability of transitioning from the current state s to state s' upon taking action a.
- γ is the discount factor from $(0, 1)$.

Notice that the reward and transition probabilities are dependent only on the current state and not on the history of states visited. This is known as the *Markov assumption*. We may also specify a start state s_{start} to denote the initial state of the agent. From this state the agent takes a series of actions a_{start}, a_1, a_2, $\ldots$, visiting states s_{start}, s_1, s_2, $\ldots$ while receiving rewards $R(s_{start})$, $\gamma R(s_1)$, $\gamma^2 R(s_2)$, $\cdots$. We can express the agent's utility as the sum of discounted rewards, $R(s_{start}) + \sum_{k=1}^{\infty} \gamma^k R(s_k)$.

Example 1. Child walking home from school

A child gets off school and needs to decide where to go. We can model his problem using a MDP M = $\{S, A, R, P, \gamma\}$. Figure 2.1(a) gives a pictorial view of the child's state space S. He starts at school in the upper left hand corner, and may choose to walk on the road toward the park on his right, or to walk towards his house at the bottom right corner. At any period in time, the child may try to move in legal horizontal or vertical direction (staying within bounds) or choose to stay put. The child will move with probability 1 if he chooses a legal direction to move in, or else stay exactly where he is. The child prefers rewards now to rewards later and has a discount factor $\gamma = 0.7$.

Figure 2.1(b) shows the child's reward function R. He does not want to stay at school, and has a reward of -1 in the start state. He does not enjoy being on the road, and has a reward of -1 for these states. He has a reward of +1 for being at the park, and a reward of +3 for being at home. If the child chooses to move to the right towards the park and stay there, we can compute his sum of discounted rewards:

$$\begin{aligned} & R(school) + \gamma R(road) + \gamma^2 R(park) + \gamma^3 R(park) + \cdots \\ = \; & 0 + 0.7(-1) + 0.7^2(1) + 0.7^3(1) + \cdots \\ = \; & -0.7 + 0.7^2(1 + 0.7 + 0.7^2 + \cdots) \\ = \; & -0.7 + .49(3.33) \\ = \; & 0.93 \end{aligned}$$

s	r	p	r
r	r	r	r
r	r	r	r
r	r	r	h

(a) state space

-1	-1	1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	3

(b) reward function

$\Rightarrow$	$\Rightarrow$	$\odot$	$\leftarrow$
$\rightarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
$\rightarrow$	$\rightarrow$	$\rightarrow$	$\downarrow$
$\rightarrow$	$\rightarrow$	$\rightarrow$	$\odot$

(c) optimal policy

Figure 2.1: Example MDP of a child walking home from school, γ = 0.7

2.1.2 Properties

Now that we have defined the MDP formalism, we turn to look at the agent's decision problem. We assume that the agent is rational, in that the agent maximizes his expected utility by choosing actions to maximize the expected sum of discounted rewards. The agent's decisions form a policy, which describes the agent's actions in every state of the world at any time period. We restrict our attention to stationary policies:[1]

Definition 2. A *stationary policy* is a function π from states to actions, such that $\pi(s)$ is the action the agent executes in state s, regardless of the time period.

Given a stationary policy π, we can define the value function V^π:

Definition 3. A *value function* V^π from S to $\mathbb{R}$ represents the total sum of discounted rewards from state s to the infinite future under policy π:

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P(s, s', \pi(s)) V^\pi(s') \tag{2.1}$$

Notice that the value function is defined recursively, such that the value in state s is the reward in state s plus the discounted expected value from possible transitioned to states s' while following policy π.

Definition 4. An *optimal policy* π^* chooses the action that maximizes the value function in every state:

$$V^{\pi^*}(s) = \max_{a \in A} R(s) + \gamma \sum_{s' \in S} P(s, s', a) V^{\pi^*}(s'), \forall s \in S \tag{2.2}$$

For each action a considered in state s, we define the value of taking action a and following the optimal policy in future states as the Q function:

[1] We can do this without loss of generality in the sense that an agent will always have an optimal stationary policy as an optimal policy. We can easily extend our model to allow for other policies, but the current model should suffice for our purposes.

Definition 5. The *Q function* from $S \times A$ to $\mathbb{R}$ satisfies the following equation:

$$Q(s,a) = R(s) + \gamma \sum_{s' \in S} P(s,s',a) V^{\pi^*}(s'), \forall s \in S \tag{2.3}$$

We can rewrite Equation 2.2 in terms of the Q function:

$$V^{\pi^*}(s) = \max_{a \in A} Q(s,a) \tag{2.4}$$

One way to solve this set of equations is to use a linear program (LP). Consider the following formulation:

$$\min_{V,Q} \sum_{s} c(s) \cdot V(s) \tag{2.5}$$

subject to:

$$V(s) \geq Q(s,a) \quad \forall s \in S, a \in A \tag{2.6}$$

$$Q(s,a) \geq R(s) + \gamma \sum_{s' \in S} P(s,s',a) V(s') \quad \forall s \in S, a \in A \tag{2.7}$$

where $c(s)$ are positive constants such that $\sum_s c(s) = 1$.[2] The constraints ensure that $V(s)$ is an upper bound for the value of the MDP [20], and the objective pushes $V(s)$ down to be exactly that value. Using the solved Q values, the optimal policy π^* is given by $\pi^*(s) \in \arg\max_a Q(s,a)$. Notice that there may be more than one optimal policy, since there may be more than one action that is optimal for a given state.

Figure 2.1(c) shows an optimal policy for the child in Example 1. The arrows in the state denote the direction the child will move in for each state, and $\odot$ denotes the child choosing to stay put in the state. We see that starting from school in the upper left corner, the child will walk right towards the park, and choose to stay at the park and not go home. While the child gets a larger reward for being home, the costs on the path towards home are too high for the reward to be worth going for, unless the child was already within two steps of home.

[2]Actually, any positive numbers will work for c(s). Having them sum to 1 allows for $c(s)$ to represent a probability distribution over possible start states.

2.2 Policy Teaching with Known Rewards

Now that we have introduced Markov Decision Processes, we can start to define the policy teaching problem within the MDP framework. But first, what is it that the interested party wants from the agent? How do we think about the external goal?

2.2.1 The Interested Party's Problem

We begin to answer these questions by considering the interested party's problem. Since the interested party cares about the agent's behavior in the domain, we can say that the interested party faces the same problem as the student, but with rewards that represent the interested party's preferences over the state of the world.

Definition 6. An *MDP\R* is a quadtuple $M_{-R} = \{S, A, P, \gamma\}$ that represents the state space, action space, probability transition, and discount factor of the problem domain.

Given an agent facing a MDP $M = \{S, A, R, P, \gamma\}$, we can describe the problem domain with a MDP\R $M_{-R} = \{S, A, P, \gamma\}$. We can then describe the interested party's problem with a MDP $T = M_{-R} \bigcup \{G\}$, where G is the interested party's reward function. We can solve for the set of optimal policies Π_T with respect to the interested party's MDP T. If the agent performs a policy $\pi_T \in \Pi_T$, the interested party is satisfied and the policy teaching problem is solve.

However, it is unreasonable to believe that the agent will perform a policy π_T. After all, the agent does not face the same reward as the interested party. The reward function R of the agent will most likely bear some similarities with the interested party – the student has positive value for getting problems correctly, the child has positive value for being home, and the consumer has positive value for the good. However, an agent has costs and rewards that are not taken into account by the interested party's preferences. To take the child walking home example,

the child may undervalue being home (as compared to the parents), have costs for walking on the road, and have rewards for being at the park. Given these factors, the child's default optimal policy is unlikely to simultaneously be a policy that performs optimally for the parents' problem.

We can denote the agent's personal costs and rewards by a function C, and write the agent's reward as $R = G + C$. The interested party can provide an *incentive function* Δ, such that the agent's modified reward function $R' = G + C + \Delta$. We define the notion of *admissible incentive function* to express the limits on the incentives an interested party provides:

Definition 7. An incentive function Δ is *admissible* if it satisfies the following constraints:

- $\Delta(s) \geq 0, \forall s \in S$

  We want to restrict our attention to only positive incentives, that is, we do not allow the interested party to use punishment to affect the agent's behavior.

- $\sum_s \Delta(s) \leq D_{max}$

  The amount of incentives the interested party is willing to provide is bounded by D_{max}.[3]

One seemingly simple thing for the interested party to do is just to provide a motivating reward $\Delta = -C$ to induce an agent reward $R' = G$ based on which the agent will behave as the interested party desired. However, there are a couple of problems with this idea. First, if the agent has any positive personal reward for some state s such that $C(s) > 0$, then the interested party would have to provide a punishment $\Delta(s) < 0$, which violates the admissible condition. Second, $\sum_s \Delta(s)$

[3]This constraint is not exactly accurate. Since an agent may visit a state more than once, limits on the incentives provided in states should be dependent on the number of times each state is visited. Since we do not know a priori what states the agent will visit and how many times, $\sum_s \Delta(s)$ provides a rough estimate of the amount of incentives that will be provided. We can get a better estimate by including history into the state space, but this requires an exponential blowup of the state space. We will return to this issue in Chapter 5.

may be greater than the maximum motivation D_{max} that the interested party is willing to provide. Finally, in the teaching, parenting, and shopping examples that motivate our work, we would imagine that the interested party hopes that an agent eventually "absorbs" the provided incentives; that is, the agent over time internalizes the provided incentive and performs as the interested party desires without having to provide any (or much) external incentives. If Δ is large, the agent may not be able to motivate himself enough to internalize the provided incentive.[4]

Nevertheless, there may be some cases in which the interested party may be able to provide a $\Delta = -C$. More generally, the interested party aims to provide incentives such that the agent performs well under the interested party's problem, even if the agent's policy is not optimal under the interested party's MDP T. We see an example of this below.

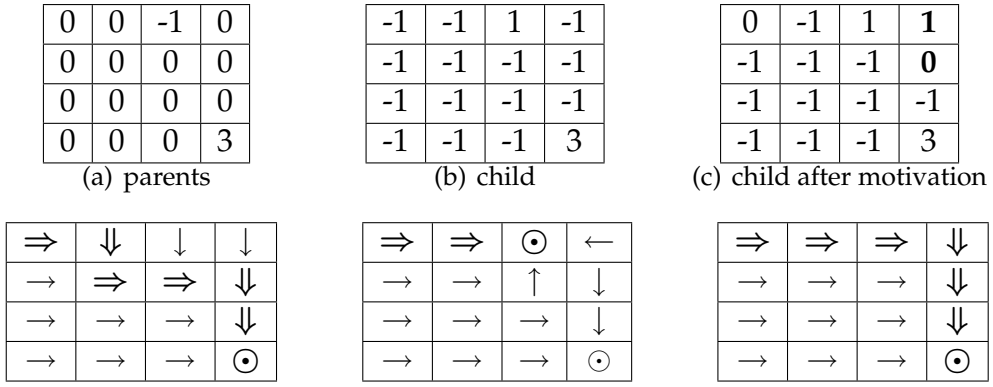


Figure 2.2: Example 2. Figure shows the reward functions and optimal policies for the parents', child's, and motivated child's problems.

Example 2. Child walking home, continued.

We expand Example 1 to illustrate the interested party's (parents') problem, the agent's (child's) problem, and the agent's problem after provided incentives. In Figure 2.2(a), we see that the parents have a small negative reward for the child going to the park, and has a high positive reward for the child coming home. The

[4]There are also a number of pedagogical and psychological issues involved, which we discuss in Chapter 5.

optimal policy for the parents' problem is for the child to take the shortest path home while avoiding the park. In Figure 2.2(b), however, we see that the child has positive reward for going to the park, undervalues going home, and has a cost for being on the road. The optimal policy for the child is to go to the park and stay there. In Figure 2.2(c), we see that the parents have provided the child with incentives in the two upper rightmost squares. The child will still visit the park, but now has enough incentive to move to the right and then down until he reaches home. Notice that the parents' provided incentives were enough to get the child to go home, but not enough to keep the child away from the park. While the motivated child does not perform optimally with respect to the parents' problem, the parents are much happier that the child comes home than stay in the park.

Figure 2.2(c) shows that even with limited incentives, when incentives of the right amounts are placed in the correct states, the interested party can induce the agent to perform a desirable policy. Figuring out what incentive function to provide will be the focus of our attention.

2.2.2 Setup

Now that we have described the interested party's problem, we define the general policy teaching problem with known reward.

Definition 8. Policy teaching with known rewards
An agent faces a sequential decision task modeled by a MDP $M = \{S, A, R, P, \gamma\}$ and performs the optimal policy π_{agent}. An interested party knows the agent's MDP M, observes the agent's policy, and wishes for the agent's policy to "perform well" under the interested party problem modeled by a similar MDP $T = \{S, A, G, P, \gamma\}$. The interested party can provide an incentive Δ to modify the agent's reward function to $R' = R + \Delta$. Is there an admissible Δ for which the agent's induced optimal policy π'_{agent} with respect to the modified MDP $M' = \{S, A, R', P, \gamma\}$ "performs well" with respect to T?

We see that translating the problem statement to the MDP setting is fairly natural. Notice that the conditions on π'_{agent} performing well in the interested party's problem are purposely left out of Definition 8, and will be specified for the specific domains we consider. In the problem definition we have restricted the agent's modified rewards to be additive; this is easily generalizable to any affine transformation over the incentive function.[5] To focus our attention, we make the following additional assumptions:

Assumption 1. The state and action spaces are finite.

Assumption 2. $R(s) \leq R_{max} \quad \forall s \in S$
The agent's reward function is bounded.

These assumptions are not very restrictive. More difficult to satisfy is the interested party having knowledge of the agent's reward function. Nevertheless, certain conditions can make this possible. First, because we are in a repeated setting, the interested party may have learned this information over time, i.e. from using a direct preference elicitation mechanism to query for the agent's preferences. Second, the interest party may have learned the agent's reward function based on observations of the agent's behavior in multiple settings. Third, in some cases, the agent may be able articulate his preferences and the interested party may learn of these preferences from the agent sharing the information directly. One possible example of this is shopping for clothing, where a consumer may let the salesperson know of her values for different types of clothing.

Techniques for solving for policy teaching in settings with known rewards will still be useful when we delve into settings with unknown rewards. But how do we approach this problem at all? To induce the agent to perform a policy π' that performs well in the interested party's problem, the interested party must find

[5]The interested party would have to know the affine transformation. In domains we consider in which agents have repeated interactions with the interested party, the interested party can learn the transformation over time. Furthermore, the effect of incentives on an agent is likely to be (almost) independent of state, which simplifies the learning problem.

some $R' \geq R$ under which the agent performs π'. We can do this with a simple algorithm sketch:

1. If $\pi = \pi'$, do nothing (the agent is already behaving as desired).
2. Otherwise, find the least $R' \geq R$ that induces the agent to perform policy π'.
3. If $\Delta = R' - R$ is admissible, return success. Otherwise, return failure.

Finding the least $R' \geq R$ that induces π' ensures that we find an admissible incentive function if one exists. The admissible requirement is important because given Assumption 2 and enough incentives, it is always possible to induce the agent to perform optimally in the interested party's problem. But up until now, we have yet to discuss what policy π' would perform well in the interested party's problem. How do we find this π'?

2.2.3 Domains without an Expert

Now that we have defined the general policy teaching problem, we must define what it means for the agent to do well under the interested party's MDP. One natural way to define doing well is as follows:

Definition 9. Goal in domains without an expert
Given the agent's reward function and the constraints on Δ, the agent performs well in the interested party's problem if the agent's motivated rewards induce a policy π' that maximizes the interested party's value function in the start state.

We can restate the policy teaching problem with this goal in mind:

Definition 10. Policy teaching in domains without an expert
An agent faces a MDP $M = \{S, A, R, P, \gamma\}$. An interested party faces a MDP $T = \{S, A, G, P, \gamma\}$, and wishes for the agent's optimal policy to perform well under T. The interested party knows the agent's MDP M and observes the agent's policy,

and can provide an incentive function Δ to modify the agent's reward function to $R' = R + \Delta$. Denoting the interested party's value function by GV, for what admissible Δ can $R + \Delta$ induce a policy π' that maximizes $GV(s_{start})$?

We can attempt to solve this problem by forming a linear program as we did for finding an optimal policy. However, unlike solving for the V and Q functions and figuring out the optimal policy from the results of the optimality program in Equation 2.5, we need to know the agent's optimal policy within the program to be able to use it to define the interested party's value function under the agent's policy. We can accomplish this by using a mixed integer program (MIP), in which we define integer indicator variables $x_{si} \in \{0, 1\}$ which tells us whether the agent takes action i in state s. With known rewards, we can solve the policy teaching problem stated in Definition 10 using the following formulation:

$$\max_{\Delta} \; GV(s_{start}) \tag{2.8}$$

subject to:

$$Q(s,a) = R(s) + \Delta(s) + \gamma \sum_{s'} P(s,s',a)V(s') \qquad \forall s,a \tag{2.9}$$

$$V(s) \geq Q(s, a_{si}) \qquad \forall s,i \tag{2.10}$$

$$V(s) \leq M_v(1 - X_{si}) + Q(s, a_{si}) \qquad \forall s,i \tag{2.11}$$

$$GQ(s,a) = GR(s) + \gamma \sum_{s'} P(s,s',a)GV(s') \qquad \forall s,a \tag{2.12}$$

$$GV(s) \geq -M_{gv}(1 - X_{si}) + GQ(s, a_{si}) \qquad \forall s,i \tag{2.13}$$

$$GV(s) \leq M_{gv}(1 - X_{si}) + GQ(s, a_{si}) \qquad \forall s,i \tag{2.14}$$

$$\sum_{s} \Delta(s) \leq D_{max} \tag{2.15}$$

$$\Delta(s) \geq 0 \qquad \forall s \tag{2.16}$$

$$\sum_{i} X_{si} = 1 \qquad \forall s \tag{2.17}$$

$$X_{si} \in \{0, 1\} \qquad \forall s,i \tag{2.18}$$

Constants $M_v = \overline{M_v} + |\underline{M_v}|$ and $M_{gv} = \overline{M_{gv}} + |\underline{M_{gv}}|$ are set such that $\overline{M_v} = (\max_s R(s) + D_{max})/(1-\gamma)$, $\underline{M_v} = \min_s R(s)/(1-\gamma)$, $\overline{M_{gv}} = \max_s GR(s)/(1-\gamma)$, and $\underline{M_v} = \min_s GR(s)/(1-\gamma)$. Constraint 2.9 defines the agent's Q functions in terms of R and Δ. Constraints 2.10 and 2.11 ensure that the agent takes the action with the highest Q value in each state. To see this, consider the two possible values for X_{si}. If $X_{si} = 1$, $V(s) = Q(s, a_{si})$. By Constraint 2.10, $Q(s, a_{si}) = \max_i Q(s, i)$. If $X_{si} = 0$, the constraints are satisfied because $M_v \geq \max V(s) - Q(s, a_{si})$.[6] Constraint 2.12 defines the interested party's Q function GQ in terms of GR. Constraints 2.13 and 2.14 ensure that the interested party's value in a state is equal to the Q value of the action from the agent's policy. Constraints 2.15 and 2.16 ensure that Δ is admissible, and Constraints 2.17 and 2.18 ensure that exactly one action is chosen for each state. The objective maximizes the interested party's value in the start state.

While the program presented above solves the policy teaching problem stated in Definition 10, the formulation requires the solution to a mixed integer linear program, which is an NP-hard problem. Whether the policy teaching problem in Definition 10 is NP-hard is an open question. Furthermore, while mixed integer programs can sometimes be solved efficiently, using big-M constants increases the size of the search space and makes solving MIPs more computationally expensive. In our formulation, we have chosen the values of M_v and M_{gv} tightly in an attempt to control the size of the search space.[7] Nevertheless, finding better formulations or using other approaches for solving the policy teaching problem may be necessary for some settings.

[6] Since $\overline{M_v}$ is the sum of discounted rewards for staying in the state with the highest possible reward, and $\underline{M_v}$ is the sum of discounted rewards for staying in the state with the lowest possible reward, it must be that $\overline{M_v} \geq \max V(s)$ and $\underline{M_v} \leq \min Q(s, a_{si})$. This implies that $M_v \geq \max V(s) - Q(s, a_{si})$.

[7] Slightly tighter bounds are possible by defining big-M constants for each state. However, further tightening will require finding the maximal value achievable in a state given Δ, which is itself a computationally expensive process (that can be solved by a mixed integer program similar to Program 2.8).

2.2.4 Domains with an Expert

In many situations, there may be an expert in the problem domain who can guide the agent to do well in the interested party's problem. The expert may have intrinsic rewards similar to the agent's, and has been motivated either by himself, the interested party, or another expert to perform well under the interested party's problem. In the teaching setting, this could be an older student, who once faced the same struggles as the student but over time has been motivated to solve problems correctly. In the child walking home setting, the expert could be an older brother, who once desired to stay at the park just like the child but now knows the importance of going home. In the shopping example, the expert may be a friend who did not want to make a purchase before, but has since made a purchase due to discounts.

Definition 11. In a problem domain defined by a MDP\R $M_{-R} = \{S, A, P, \gamma\}$, an *expert* faces a MDP E = $\{S, A, R_E, P, \gamma\}$ and has an optimal policy π_E which "performs well" in the interested party's problem.

Given an expert, instead of solving an optimization problem on the value function in the start state of the interested party's problem, the interested party needs only to provide incentives such that the agent behaves like the expert:

Definition 12. Goal in domains with an expert
Find an admissible Δ such that the agent's modified reward $R' = R + \Delta$ induces the expert's policy π_E.

While this goal is clear, it is not always reachable. For example, consider the case when the expert has the same reward function as the interested party. The expert's policy is optimal for the interested party's problem, but there is unlikely to be an admissible incentive function that can modify the agent's reward to induce the expert's policy. For the expert to be useful for policy teaching, the expert must

both perform well in the interested party's problem (i.e. have a high value in the start state) and perform a policy reachable by the agent given the constraints on Δ.

We can think of the expert as having faced a similar reward as the student and has since been motivated to perform well in the interested party's domain. We can view the expert's reward $R_E = R_O + \Delta_E$ as the sum of the expert's intrinsic reward R_O and the motivation Δ_E that he received. Given that R_O is similar to the agent's reward and Δ_E is within the constraints on incentives, we can think of the expert as having solved a similar policy teaching problem.

Now that we have defined the goal and have discussed the conditions for the goal to be reachable, we can define the policy teaching problem for domains with an expert:

Definition 13. Policy teaching in domains with an expert
An agent faces a MDP $M = \{S, A, R, P, \gamma\}$. An interested party faces a MDP $T = \{S, A, G, P, \gamma\}$, has knowledge of the agent's MDP M, observes the agent's policy, and can provide an incentive function Δ to modify the agent's reward function to $R' = R + \Delta$. The interested party also has knowledge of an expert's MDP E = $\{S, A, R_E = R_O + \Delta_E, P, \gamma\}$ and observes the expert's policy π_E. For what admissible Δ does R + Δ induce the agent to perform π_E?

Instead of solving an optimization problem on the value function in the start state of the interested party's problem, we can solve the policy teaching problem in domains with an expert if we can find a mapping from the agent's reward function to the expert's reward function. If $R_O = R$, then the interested party can simply provide the agent with Δ = $R_E - R$ = Δ_E. The agent's modified reward $R' = R_E$ induces the expert's policy π_E. However, there are a couple of issues with this:

- **R_O may be different from R.**
  It is unreasonable to expect the expert's intrinsic reward to be exactly identical to the agent's reward. It is more reasonable to expect the expert's intrinsic

reward to be similar to the agent's current reward.

- **The expert may have used more motivation than necessary to achieve $\pi_{\mathbf{E}}$.** There may be smaller incentives that the interested party can provide to induce π_E than the motivation the expert had used. The expert could have arrived at his current state through a lot of internal and external motivation, and it may be unnecessarily costly to provide the same to the agent.

We make two simple observations to deal with these issues. One observation is that we do not really care whether $R_O = R$, but instead care about whether the expert's motivated reward R_E is one that the agent can reach given the constraints on Δ. In other words, if there exists a $\Delta = R_E - R$ that is admissible, the interested party can provide this Δ such that the agent performs π_E. The second observation is that an agent's policy is invariant to positive linear shifts in the reward function:

Claim 1. Given a MDP $M = \{S, A, R, P, \gamma\}$ and an optimal policy π^*, π^* is also an optimal policy of any MDP $M' = \{S, A, R', P, \gamma\}$, where R' = cR for any constant $c > 0$.

Proof. We can write down the value function of M under π^* as in Equation 2.1:

$$V^{\pi^*}(s) = R(s) + \gamma \sum_{s' \in S} P(s, s', \pi^*(s)) V^{\pi^*}(s') \tag{2.19}$$

Multiplying by c, we get:

$$cV^{\pi^*}(s) = cR(s) + \gamma \sum_{s' \in S} P(s, s', \pi^*(s)) cV^{\pi^*}(s') \tag{2.20}$$

Denoting the value function under M' as V', we can write down this value function under π^*:

$$V'^{\pi^*}(s) = cR(s) + \gamma \sum_{s' \in S} P(s, s', \pi^*(s)) V'^{\pi^*}(s') \tag{2.21}$$

We see that $V' = cV$ is a solution for V'; the value function under M' is shifted by c just like the reward R'. Denoting the Q function under M' as Q', we see that the

Q function is also shifted by c:

$$Q'(s,a) = cR(s) + \gamma \sum_{s' \in S} P(s,s',a) V'^{\pi^*}(s'), \forall s \in S \quad (2.22)$$

$$Q'(s,a) = cR(s) + \gamma \sum_{s' \in S} P(s,s',a) cV^{\pi^*}(s'), \forall s \in S \quad (2.23)$$

$$Q'(s,a) = cQ(s,a) \quad (2.24)$$

Since $\pi^* \in \arg\max_a Q(s,a)$ and $c > 0$, $\pi^* \in \arg\max_a cQ(s,a)$. □

Using these observations, given an agent with reward R and an expert policy π_E based on reward R_E, solving the policy teaching problem requires only a $\Delta = cR_E - R$ to be admissible for some $c > 0$. We can aim to find the minimal admissible Δ by using the following linear program:

$$\min_c \sum_s \Delta(s) \quad (2.25)$$

subject to:

$$\Delta(s) = cR_E(s) - R(s) \quad (2.26)$$

$$c > 0 \quad (2.27)$$

$$\sum_s \Delta(s) \leq D_{max} \quad (2.28)$$

$$\Delta(s) \geq 0 \quad \forall s \quad (2.29)$$

If we find a feasible solution to this LP, we can provide the agent with minimal incentive Δ to perform π_E, even if the expert's intrinsic reward R_O is different than the agent's reward R and the expert's motivation Δ_E is larger than Δ.

This linear program is simple and does not pose the computational difficulties posed by the integer programming formulation for domains without experts. Given this, we would like to further loosen the requirements necessary for an expert to be useful. Policies are invariant to positive linear shifts to the reward function; are there more general conditions under which policies are invariant? Also, what happens if the interested party does not have access to the expert's rewards but only the expert's policy?

Chapter 3

Policy Teaching with Unknown Rewards

In this chapter we introduce inverse reinforcement learning (IRL) as a method for inferring the space of reward functions that corresponds to a policy, and use the IRL method to discuss the policy teaching problem with unknown rewards.

3.1 Inverse Reinforcement Learning

As we saw at the end of Chapter 2, better understanding of the space of rewards that induce a certain policy is useful for policy teaching. First, knowing the space of rewards that induces the expert's policy increases the likelihood of having an admissible incentive function that would induce the agent to perform the expert's policy. At the same time, it also allows for the providing of minimal incentives to accomplish the same task. To visualize this, we can think of the policy teaching problem as finding an *admissible mapping* from the agent's reward (a point) into any point in the space of rewards that induce the expert's policy. Figure 3.1 gives a pictorial depiction of this operation.

Second, the interested party may only observe the expert's policy and not the expert's reward function. For example, a teacher can see how an excellent student goes about solving a problem, but have no idea what the student's reward function

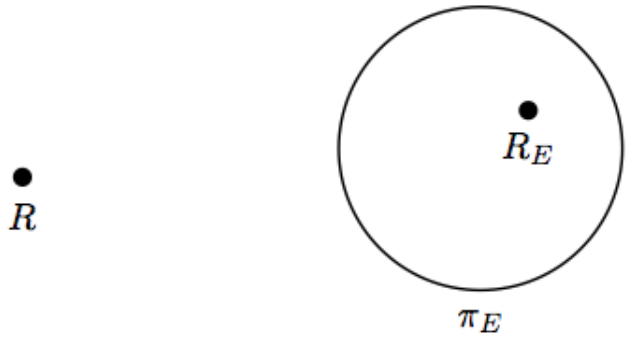


Figure 3.1: Any admissible mapping from the agent reward R to the space of rewards that induce the expert's policy π_E would solve the policy teaching problem. An interested party would prefer the shortest admissible mapping; mapping directly to the expert's reward R_E is unnecessary and may be costly.

is. Having a way to derive the expert's space of rewards based only on his policy allows us to solve the policy teaching problem even when the expert's reward function is unknown.

Finally, in many situations, the interested party may be able to observe the agent and expert's policies, but not know the reward function of the expert nor the reward function of the agent. In these cases, we need to map the agent's true reward (which is unknown) to the expect's space of rewards. If we pick an arbitrary point $\widehat{R}$ in the space of possible agent rewards as we do in Figure 3.2, the incentive Δ calculated based on this point can map $\widehat{R}$ to an expert reward $\widehat{R_E}$, but is unlikely to map the expert's true reward into the space of expert rewards.

3.1.1 Definition

Techniques from *Inverse Reinforcement Learning* allow us to learn the space of reward functions that correspond to a particular policy. The techniques rely only on the problem domain definition and the agent's optimal policy.

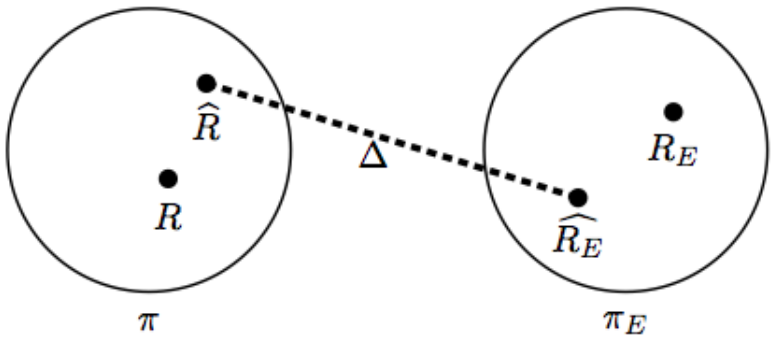


Figure 3.2: The agent faces reward R and an expert faces reward R_E. Rewards $\widehat{R}$ and $\widehat{R_E}$ are arbitary points in the space of rewards for the agent and expert policies π and π_E, respectively. An incentive $\Delta = \widehat{R_E} - \widehat{R}$ would map $\widehat{R}$ into the expert's policy space, but is unlikely to map the agent's actual reward R into the expert's space.

Definition 14. The *Inverse Reinforcement Learning* problem

In a problem domain defined by a MDP\R M_{-R}, an agent performs an optimal policy π. Find the space of rewards IRL^{π} such that $R \in IRL^{\pi}$ induces an optimal policy π.

The IRL problem is the opposite of the reinforcement learning problem of finding the optimal policy given a MDP. Reversing the process, we can place constraints on the agent's reward function based on the agent's policy. Given that we assume finite states and actions, we can express all functions in the MDP as vectors and matrices represented by bold letters. Adopting Ng and Russell [16]'s notation, for a MDP with N states and K actions, we can represent the reward and value functions as N dimensional vectors, and the Q function as an $N \times K$ matrix whose element (i, j) is the Q value of taking action j in state i. Furthermore, we use P_{π} to denote a $N \times N$ transition probability matrix whose element (i, j) is the probability of transitioning from state i to state j while taking action $\pi(i)$. We use $\prec$ and $\preceq$ to denote matrix inequalities, such that $\mathbf{x} \preceq \mathbf{y}$ if and only if $x_i \leq y_i \; \forall \; i$.

Lemma 1. *Given a MDP* $M = \{S, A, R, P, \gamma\}$, $\mathbf{V}^{\pi} = (\mathbf{I} - \gamma \mathbf{P}_{\pi})^{-1}\mathbf{R}$.

Proof. We can write the equation for V^π in matrix form as follows:

$$\mathbf{V}^\pi = \mathbf{R} + \gamma \mathbf{P}_\pi \mathbf{V}^\pi \tag{3.1}$$

$$(\mathbf{I} - \gamma \mathbf{P}_\pi)\mathbf{V}^\pi = \mathbf{R} \tag{3.2}$$

$$\mathbf{V}^\pi = (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-\mathbf{1}}\mathbf{R} \tag{3.3}$$

Equation 3.3 holds only if $(I - \gamma P_\pi)^{-1}$ is invertible. Ng and Russell [16] showed that this had to be the case because γP_π has eigenvalues on the interior of the unit circle, implying that $(I - \gamma P_\pi)$ has no zero eigenvalues and is not singular. □

Theorem 1. *Given the problem domain MDP M\R = $\{S, A, P, \gamma\}$ and an agent's optimal policy π, the agent's reward function must satisfy the following IRL constraints:*

$$(\mathbf{P}_\pi - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-\mathbf{1}}\mathbf{R} \succeq \mathbf{0} \quad \forall a \in A \tag{3.4}$$

Proof. (Ng and Russell [16])
Since π is optimal with respect to the agent's rewards, $Q(s, \pi(s)) = \max_a Q(s, a)$. We can express this as a constraint:

$$Q(s, \pi(s)) \geq Q(s, a) \qquad \forall s, a \tag{3.5}$$

$$R(s) + \gamma \sum_{s'} P(s, s', \pi(s)) V^\pi(s') \geq R(s) + \gamma \sum_{s'} P(s, s', a) V^\pi(s') \qquad \forall s, a \tag{3.6}$$

In matrix form, we have:

$$\mathbf{P}_\pi \mathbf{V}^\pi \succeq \mathbf{P_a} \mathbf{V}^\pi \quad \forall a \in A \tag{3.7}$$

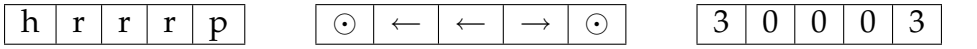
Using Lemma 1, we have:

$$\mathbf{P}_\pi (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-\mathbf{1}}\mathbf{R} \succeq \mathbf{P_a} (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-\mathbf{1}}\mathbf{R} \quad \forall a \in A \tag{3.8}$$

$$\implies (\mathbf{P}_\pi - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-\mathbf{1}}\mathbf{R} \succeq \mathbf{0} \quad \forall a \in A \tag{3.9}$$

□

The theorem uses a set of linear constraints to bound the space of rewards that corresponds to a particular policy. We give an example to illustrate how these constraints come about.



(a) state space (b) optimal policy (c) possible reward function

Figure 3.3: Simplified child walking home example, using MDP\R and optimal policy to find a potential reward function. $\gamma = 0.7$

Example 3. Simple Walk

Consider a simplified version of the child walking home example. Figure 3.3(a) shows the one dimensional state space of the child, with home at the left most end and the park at the right most end. The child may move left, right, or stay put. He moves in the direction he attempts to move with probability 0.8, and always stays still when he chooses to do so. Figure 3.3(b) shows the child's optimal policy; the child goes home and stays there if he is within two squares of home, and goes to the park and stays there otherwise.

We can figure out the space of rewards that corresponds to the child's policy. We construct probability matrices for the child's policy, moving left, moving right, and staying still, as shown in Figure 3.4. Using these matrices, Theorem 1 places IRL constraints on the reward function. Figure 3.3(c) shows a particular reward function in the space of rewards that corresponds to the child's policy. Simple matrix multiplication shows that this reward function indeed satisfies the IRL constraints.

3.1.2 Properties

Now that we have introduced techniques from IRL, we would like to apply these techniques to the policy teaching problem. In domains with experts, we can place IRL constraints on the space of rewards that induce the expert's policy. However, we immediately notice that **R = 0** always satisfies the IRL constraints, regardless of the policy. While it is not surprising that all policies have value equal to 0 when **R = 0**, this shows us that picking any point in the space of expert rewards may not

		1	2	3	4	5
S	1	1	0	0	0	0
L	2	0.8	0	0.2	0	0
L	3	0	0.8	0	0.2	0
R	4	0	0	0.2	0	0.8
S	5	0	0	0	0	1

(a) P_π

	1	2	3	4	5
1	0.8	0.2	0	0	0
2	0.8	0	0.2	0	0
3	0	0.8	0	0.2	0
4	0	0	0.8	0	0.2
5	0	0	0	0.8	0.2

(b) P_{Left}

	1	2	3	4	5
1	0.2	0.8	0	0	0
2	0.2	0	0.8	0	0
3	0	0.2	0	0.8	0
4	0	0	0.2	0	0.8
5	0	0	0	0.2	0.8

(c) P_{Right}

	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1

(d) P_{Stay}

Figure 3.4: Probability transition matrices for policy π and each action for the simplified child walking home example. In each matrix, element (i, j) gives the probability of transitioning from state i to state j on the action taken.

be restrictive enough to ensure that the motivated agent will perform the expert's policy. In Example 3, while Figure 3.3(c) gives a potential reward function for the agent's policy, we see that the reward function given can also induce a policy in which the agent moves right instead of left when in the middle square. In policy teaching, we need to ensure that the target reward strictly induces the expert's policy π_E so that an agent mapped to this point will indeed perform the expert's policy. Were we to map the agent's reward to a point in the space of expert reward functions that may also induce some other optimal policy, we risk the agent performing a policy that may not perform well in the interested party's problem.

We can deal with this issue by only considering points in the expert's reward space that strictly induce the expert's policy. We can write down strict IRL constraints on rewards R_E in the expert's space to ensure that $Q(s, \pi(s)) - Q(s, a) > 0$ in all states s for all actions a that is not $\pi(s)$:

$$(\mathbf{P}_{\pi_\mathbf{E}} - \mathbf{P}_\mathbf{a})(\mathbf{I} - \gamma \mathbf{P}_{\pi_\mathbf{E}})^{-1}\mathbf{R}_\mathbf{E} \succ \mathbf{0} \quad \forall a \in A \backslash a1, a1 \in \pi(s) \tag{3.10}$$

The use of these strict IRL constraints removes reward functions that induce more than one optimal policy from consideration, but does not prevent a reward function to be found. Given an action $\pi(s)$ from an optimal policy π, the only cases in which another action a' must necessarily be as good as a is when the state transition functions for both actions are identical in state s. In such cases, we can just model these two actions in the state as the same action.

To use these strict IRL constraints on the expert's reward in a linear program, we need to represent the constraint as a non-strict inequality. We can do this by using a small constant $\epsilon > 0$, such that the slack between the Q values of the best and second best action in a state is least ϵ. We can rewrite the constraint as follows:

$$(\mathbf{P}_{\pi_\mathbf{E}} - \mathbf{P}_\mathbf{a})(\mathbf{I} - \gamma \mathbf{P}_{\pi_\mathbf{E}})^{-\mathbf{1}} \mathbf{R}_\mathbf{E} \succeq \epsilon \quad \forall a \in A \backslash a1, a1 \in \pi(s) \tag{3.11}$$

We denote a reward function R_E that satisfies this constraint as $R_E \in \text{IRL}^{\pi_E}_{strict}$. Given an expert's policy π_E and the agent's reward R, we can now solve the policy teaching problem in domains with experts using the following linear program:

$$\min_{R_E} \sum_s \Delta(s) \tag{3.12}$$

subject to:

$$\Delta(s) = R_E(s) - R(s) \qquad \forall s \in S \tag{3.13}$$

$$(\mathbf{P}_{\pi_\mathbf{E}} - \mathbf{P}_\mathbf{a})(\mathbf{I} - \gamma \mathbf{P}_{\pi_\mathbf{E}})^{-\mathbf{1}} \mathbf{R}_\mathbf{E} \succeq \epsilon \qquad \forall a \in A \backslash a1, a1 \in \pi(s) \tag{3.14}$$

$$\sum_s \Delta(s) \leq D_{max} \tag{3.15}$$

$$\Delta(s) \geq 0 \qquad \forall s \in S \tag{3.16}$$

where R_E are variables on the target reward in the reward space of the expert's policy. The program is still a tractable LP and provides an improvement over Program 2.25 to capture policy invariance beyond just linear shifts in the reward function. Furthermore, Program 3.12 does not require knowledge of the expert's

reward function, again loosening the observability requirements on the interested party.

The use of IRL constraints have allowed us to complete our discussion of policy teaching with known rewards. However, what happens when we do not know the agent's reward? IRL gives the space of rewards that induce the agent's policy, but how do we find the agent's reward out of that space as to be able to map it to the desired space?

3.2 Policy Teaching with Unknown Rewards

3.2.1 Setup

Definition 15. Policy teaching with unknown rewards
An agent faces a MDP $M = \{S, A, R, P, \gamma\}$ and performs an optimal policy π. An interested party knows the problem definition MDP\R $M_{-R} = \{S, A, P, \gamma\}$, observes the agent's policy, but does not know the agent's reward R. The interested party wishes for the agent's policy to perform well under the interested party's problem modeled by a similar MDP $T = \{S, A, G, P, \gamma\}$, and can provide an admissible incentive Δ to modify the agent's reward function to $R' = R + \Delta$. Can the interested party learn about the agent's reward function as to be able to provide a Δ for which the agent's induced optimal policy π' with respect to the modified MDP $M' = \{S, A, R', P, \gamma\}$ performs well with respect to T?

This problem is more interesting than policy teaching with known rewards, and also more difficult. The problem is more interesting because in most situations, the interested party will not know the reward function of the agent. Even in cases in which an agent does not mind sharing his preferences with the interested party, the agent may not be able to articulate his reward function despite being able to perform optimally with respect to his preferences.

The problem is also much harder than policy teaching with known rewards.

Techniques from IRL can give us the space of possible rewards that induce the agent's policy, but how do we find the agent's actual reward within this space? Without further knowledge of the agent's actual reward, one reasonable criteria is to prefer reward functions whose corresponding Q values have a large slack, that is, the optimal action in each state is much better than the second best action. Furthermore, we may wish to penalize large rewards, both to prefer simpler reward functions and also to ensure that the large slack is not only due to the large size of the reward functions. Based on the agent's policy π, we can choose R based on this criterion using the following linear program:

$$\max_R [\sum_{i=1}^{N} \beta(s) - \lambda \sum_{i=s}^{N} \alpha(s)] \tag{3.17}$$

$$\alpha(s) \geq R(s) \qquad \forall s \in S \tag{3.18}$$

$$\alpha(s) \geq -R(s) \qquad \forall s \in S \tag{3.19}$$

$$((\mathbf{P}_\pi - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}\mathbf{R})[s] \geq \beta(s) \quad \forall a \in A \forall s \in S \tag{3.20}$$

$$(\mathbf{P}_\pi - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}\mathbf{R} \succeq \mathbf{0} \quad \forall a \in A \tag{3.21}$$

Constraints 3.18 and 3.19 ensure that $\alpha(s)$ is at least as large as the absolute value of $R(s)$ in state s. Constraint 3.20 ensures that $\beta(s)$ is at most the slack in Q value between the optimal action $\pi(s)$ and the next best action in state s. Constraint 3.21 places IRL constraints on the agent's reward function. Notice that we do not require strictness here, since the agent's true reward may induce other optimal policies in addition to π. The objective maximizes the slack on Q values while penalizing large rewards weighted by a constant $\lambda > 0$. The objective will drive $\alpha(s)$ to be its minimum value (absolute value of rewards) and drive $\beta(s)$ to its maximum value (the slack in Q values).

Ng and Russell [16] first introduced this optimization problem and Abbeel and Ng [1] used a similiar objective for apprenticeship learning. In both of these cases, the derived reward function was meant to be used by a machine agent, where hav-

ing large slack and simple reward functions allow the machine agent to perform well even with slight changes to the problem domain. In our case, the derived reward function needs to resemble the agent's true reward, based on which we can find the incentive function to map the agent's reward to the reward space of the expert's policy. Applied to policy teaching, Program 3.17 gives us a heuristic for picking a point in the agent's reward space, but otherwise does little to provide us more information on the agent's true reward.

In order to make progress towards learning the agent's reward function, it is necessary to be able to narrow the space of possible agent rewards by eliminating rewards in the space that are not the agent's actual reward. If we can gain additional evidence about the agent's reward, we may be able to use the evidence to further constrain the space of possible rewards.

3.2.2 Elicitation Method

One way to gain new evidence about the agent's reward function is for the interested party to perform direct preference elicitation by asking queries about the agent's preferences. Queries often considered in the literature [22] include rank queries ("What is your second-most preferred policy?"), order queries ("Is policy a preferred to policy b?"), bound queries ("Is your reward in state s at least r_s?"), and value queries ("what is your reward in state s?"). Responses to queries can be used to introduce additional constraints on the agent's reward function (in addition to IRL constraints), which in turn narrows the space of rewards.

While direct preference elicitation have been used successfully in many settings, there are a number of issues that suggest that direct querying may be infeasible in the sequential decision making domains we consider:

- **Interconnected states.**

  The quality of an action in a state is dependent on the rewards and actions in other states. Asking queries on the reward in a certain state requires the

agent to place explicit weights on individual states and consider a state in isolation of other states and actions. While reward functions are succinct representations of agent preferences in MDPs, it is not clear that an agent can express his preferences in this way given the interdependencies between different states and actions.

- **Policy-based queries are hard to answer.**
  While order queries comparing the optimal policy to any other policy are trivial to answer, it is not clear that rank and order queries on non-optimal policies are easy to answer. In a setting with N states and K actions, there are K^N possible stationary policies, most of which the agent probably has never considered explicitly.

- **Nonmonetary rewards are hard to articulate.**
  In many of the examples that we are interested in, the agent's preferences are nonmonetary. Furthermore, the incentives provided may also be nonmonetary, and could be of a different denomination than the agent's internal preferences. For bound and value queries, the agent would need to be able to express his preferences in the same denomination as the incentives provided.

- **Getting the agent to answer queries may be difficult.**
  In monetary situations such as auctions, getting an agent to participate in an elicitation process is "simple" - the agent must participate in order to participate in the auction. In policy teaching problems, it is not clear how the interested party can get the agent to answer queries.

Due to these issues, we consider instead methods of *indirect elicitation*, that is, using observations of an agent's behavior to make inferences about his preferences. With IRL, we are already doing indirect elicitation by using the agent's policy to find the space of reward functions that correspond to that policy. With one policy, however, we only have one set of IRL constraints. How can we generate more

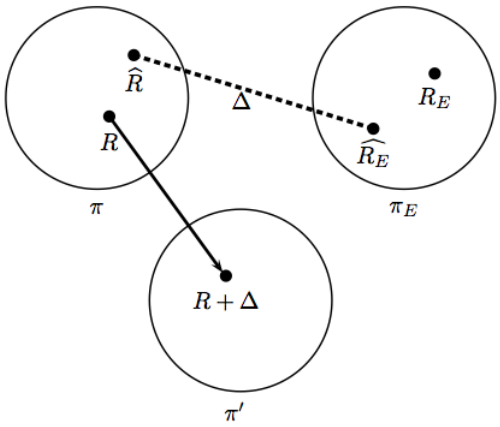


Figure 3.5: Policy teaching based on agent reward guess $\widehat{R}$ and an arbitary point R_E in the expert's reward space. An attempt to map the agent's reward R to the expert's space using $\Delta = \widehat{R_E} - \widehat{R}$ was unsuccessful. The agent's motivated reward $R + \Delta$ induced a policy $\pi' \neq \pi_E$.

constraints on the agent's reward? We can make a guess $\widehat{R}$ at the agent's reward R by choosing any point within the IRL space of the agent's policy (either randomly, using Program 3.17, or by some other means). To figure out if we have made the right guess, we can assume that the agent's true reward is $\widehat{R}$ and treat the problem as policy teaching with known rewards. In a domain with an expert like the one shown in Figure 3.5, we can find a point $\widehat{R_E}$ in the expert's reward space, to which we have an admissible Δ from $\widehat{R}$. If the agent's true reward were $\widehat{R}$, we would expect providing the agent with incentive Δ to induce the agent to perform the expert's policy π_E. If instead the agent performs a policy $\pi' \neq \pi_E$, we know that $\widehat{R_E}$ must not be the agent's true reward. Furthermore, we know that the $R + \Delta$ must induce π', which may eliminate other points in the space of agent rewards. Figure 3.6 provides a pictorial depiction of a failed mapping attempt.

Using the observation of the agent's policy π' in response to the provided in-

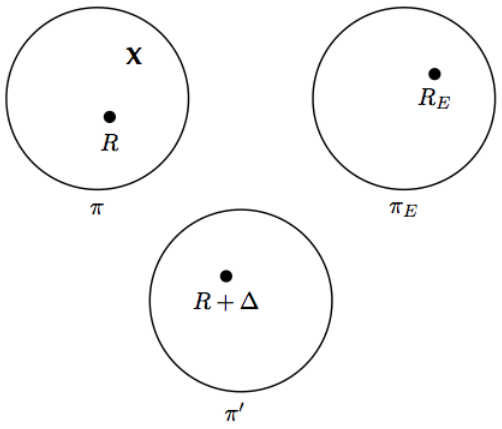


Figure 3.6: Result of a failed mapping attempt. We now know that $\widehat{R}$ cannot be the agent's true reward. We also know that $R+\Delta$ must induce policy π', which further constrains the set of rewards that could be the agent's reward.

centive, we can write down an IRL constraint on $R + \Delta$ such that $(R + \Delta) \in \text{IRL}^{\pi'}$:

$$(\mathbf{P}_{\pi'} - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P}_{\pi'})^{-\mathbf{1}}(\mathbf{R} + \mathbf{\Delta}) \succeq \mathbf{0} \quad \forall a \in A \text{ (3.22)}$$

$$\implies (\mathbf{P}_{\pi'} - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P}_{\pi'})^{-\mathbf{1}}\mathbf{R} + (\mathbf{P}_{\pi'} - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P}_{\pi'})^{-\mathbf{1}}\mathbf{\Delta} \succeq \mathbf{0} \quad \forall a \in A \text{ (3.23)}$$

We can repeat the process of guessing a reward in the agent's IRL space, mapping it to a point in the IRL space of a desired policy, observing the induced agent policy, and adding new constraints if the agent does not behave as desired. We can sketch out the basis of an algorithm:

1. Choose a potential agent reward $\widehat{R}$ such that $\widehat{R} \in IRL^{\pi_{agent}}$.
2. Find admissible mapping $\widehat{\Delta}$ to some $\widehat{R_T} \in IRL^{\pi_T}_{strict}$, where π_T is the target policy that performs well in the interested party's problem.
3. Provide agent with incentive $\widehat{\Delta}$. Observe the agent's modified policy π'.
4. If $\pi' = \pi_T$, we are done. Otherwise, add constraint $(R + \widehat{\Delta}) \in IRL^{\pi'}$.
5. Repeat.

The IRL constraints on the target policy are strict to ensure that the agent performs the desired policy when we have the right guess $\widehat{R}$ for R. Note that we learn about the agent's reward only for the purpose of finding an admissible mapping to the IRL space of the target policy. This suggests that it is possible to find a mapping to the desired policy before we have found the agent's exact reward R..

This algorithm sketch gives us a general way of thinking about policy teaching with unknown rewards, which we use to construct algorithms for policy teaching in domains with and without an expert.

3.2.3 Domains with an Expert

Definition 16. Policy teaching with unknown rewards in domains with an expert An agent faces a MDP $M = \{S, A, R, P, \gamma\}$ and performs the optimal policy π. An interested party knows the problem definition MDP\R $M_{-R} = \{S, A, P, \gamma\}$ and observes the agent's policy, but does not know the agent's reward R. The interested party also observes an expert's policy π_E, but does not know the expert's reward R_E. The interested party may provide an admissible incentive Δ to modify the agent's reward function to $R' = R + \Delta$. Can the interested party learn about the agent's reward function so as to be able to provide a Δ such that the agent performs the expert policy π_E?

The expert's policy offers a clear target IRL space into which we can attempt to map the reward of the agent. In guessing a $\widehat{R}$ for the agent's reward, we can restrict our attention to rewards in the agent's IRL space that has admissible mappings into the expert's IRL space. We can accomplish this by combining step 2 and 3 of the algorithm sketch to find $\widehat{R}$ and $\widehat{\Delta}$ simultaneously instead of guessing $\widehat{R}$ and then seeing if there is an admissible $\widehat{\Delta}$ from $\widehat{R}$ to a point in the expert's IRL space.

We adopt the following notation for our algorithm. We write IRL constraints on a reward variable R as $R \in \text{IRL}^{\pi}$. All constraints are added to a constraint set K, such that the instantiations of variables must satisfy all constraints in K (but may

take on any value as long as it does satisify the constraints in K). An instantiation of a variable R is denoted as $\widehat{R}$.

We have the following algorithm:

Algorithm 1 Elicitation Method in domains with an expert

Given policies π, π_E; variables R, R_E, Δ; constraint set $K = \emptyset$
Add $R \in \text{IRL}^{\pi}$ to K
Add $R_E \in \text{IRL}^{\pi_E}_{strict}$ to K
Add $\Delta = R_E - R$ to K
Add $admissible(\Delta)$ to K
loop
  Find values $\widehat{\Delta}$, $\widehat{R}$, $\widehat{R_E}$ that satisfies all constraints in K
  if no such values exist **then**
    return FAILURE {no possible mappings from agent to expert}
  else
    Provide agent with incentive $\widehat{\Delta}$
    Observe agent policy π' with respect to $R' = R^{true} + \Delta$.
    if $\pi' = \pi_E$ **then**
      return $\widehat{\Delta}$
    else
      Add $(R + \widehat{\Delta}) \in \text{IRL}^{\pi'}$ to K
    end if
  end if
end loop

Algorithm 1 gives a general elicitation method in domains with experts; any objective on what values to choose for the variables can be used as long as the values satisfy the constraints in K (which are all linear). We can show that the elicitation process makes progress at every iteration:

Lemma 2. *In each iteration of the loop in Algorithm 1, the added constraint* $(R + \widehat{\Delta}) \in$ *IRL*$^{\pi'}$ *removes at least one point from the space of possible agent rewards.*

Proof. Since $\widehat{R} + \widehat{\Delta} = \widehat{R_E}$ and $\widehat{R_E}$ strictly induces an optimal policy π_E, $(\widehat{R} + \widehat{\Delta}) \notin$ IRL$^{\pi'}$. The added constraint $(R+\widehat{\Delta}) \in$ IRL$^{\pi'}$ eliminates $\widehat{R}$ from the space of possible agent rewards. □

Theorem 2. *If $R(s)$ is discrete over $\mathbb{R}$ for all s, Algorithm 1 terminates in a finite number of steps with an admissible mapping Δ, or returns FAILURE if no such mapping exists.*

Proof. By Assumption 2, $R(s)$ is bounded. If $R(s)$ is also discrete, $R(s)$ must be finite. Since the state space is finite by Assumption 1, there must only be a finite number of possible agent reward functions. By Lemma 2, either one of these reward functions is eliminated in each iteration or the elicitation process terminates. This implies that in a finite number of iterations, the algorithm either finds an admissible mapping or determines that no admissible mapping is possible when all possible rewards have been eliminated. Since each iteration takes a finite number of steps, the elicitation process must terminate in a finite number of steps. □

When the space of possible agent rewards is discrete (or has been discretized), Theorem 2 ensures completion of the elicitation process within a finite number of steps. Generally, we expect the agent's reward function to arise from a continuous spectrum; considering a discrete space of possible agent rewards may be too restrictive to capture the agent's true reward. The following lemmas make progress towards a more general theorem for non-discrete agent rewards:

Lemma 3. *Given a MDP $M = \{S, A, R, T, \gamma\}$, we can construct a perturbed MDP $M' = \{S, A, R', T, \gamma\}$ such that $|R'(s) - R(s)| \leq \delta$, $\forall s \in S$, $\delta > 0$. Denoting Q as the Q function for M and Q' as the Q function for M', $\forall s \in S, a \in A$:*

$$|Q'(s,a) - Q(s,a)| \leq \frac{\delta}{1-\gamma} \tag{3.24}$$

Furthermore, $\forall s \in S, a, a' \in A$, we can bound the change in slack by:

$$|(Q'(s,a) - Q'(s,a')) - (Q(s,a) - Q(s,a'))| \leq \frac{2\delta\gamma}{1-\gamma} \tag{3.25}$$

Proof. To cause the largest perturbation to the Q value, we must perturb the reward in every state in the same direction. Since the reward function is perturbed by at most δ in each state, the greatest series of perturbations that can contribute to

an agent's Q value is $\sum_{k=0}^{\infty} \delta\gamma^k = \frac{\delta}{1-\gamma}$. To cause the largest perturbation to the difference between the Q values of two actions in the same state, we can give the greatest series of future perturbations to one action and the least series of future perturbations to the other action, bounding the change in difference by $2\sum_{k=0}^{\infty} \delta\gamma^k = \frac{2\delta\gamma}{1-\gamma}$. □

Lemma 4. *Based on an agent's reward guess $\widehat{R}$ and proposed mapping $\widehat{\Delta}$, we can construct a MDP M' = $M_{-R} \bigcup \{\widehat{R} + \widehat{\Delta}\}$. We calculate the Q functions with respect to M', and denote the slack in Q values by $\epsilon = \min Q(s, \pi_E) - Q(s, a_s)$, $\forall s \in S, a_s \in A \backslash \pi_E(s)$. If $\widehat{R}$ is eliminated from the agent's space of rewards, it must be that no points in the open hypercube $C = \{x : \widehat{R}(s) - \frac{\epsilon(1-\gamma)}{2\gamma} < x(s) < \widehat{R}(s) + \frac{\epsilon(1-\gamma)}{2\gamma}, \forall s \in S\}$ can be the agent's reward.*

Proof. Since $\widehat{R} + \widehat{\Delta}$ strictly induce the expert's policy, it must be that $\epsilon > 0$. Up to ϵ, we can perturb the Q values in any state without changing the optimal policy. By Lemma 3, we know we can perturb $\widehat{R} + \widehat{\Delta}$ by δ in any number of states without changing the optimal policy as long as $\frac{2\delta\gamma}{1-\gamma} < \epsilon$. Solving the inequality for δ, we see that all points in an axis-aligned open hypercube with side length $\delta = \frac{\epsilon(1-\gamma)}{2\gamma}$ centered at $\widehat{R} + \widehat{\Delta}$ must strictly induce the expert's policy π_E. This implies that there exists an axis-aligned open hypercube of the same size around $\widehat{R}$ such that there is a one-to-one mapping from this hypercube to the hypercube centered at $\widehat{R} + \widehat{\Delta}$ through $\widehat{\Delta}$. Since $R^{true} + \widehat{\Delta}$ induces a policy $\pi' \neq \pi_E$ when $\widehat{R}$ is eliminated, it must be that no points in the open hypercube $C = \{x : \widehat{R}(s) - \frac{\epsilon(1-\gamma)}{2\gamma} < x(s) < \widehat{R}(s) + \frac{\epsilon(1-\gamma)}{2\gamma}, \forall s \in S\}$ can be the agent's reward. □

Lemma 4 bounds a volume around each eliminated point such that every point within the volume must not be the agent's reward. Figure 3.7 provides a visual representation of this process. We pick $\widehat{R}$ from the IRL space of π_{agent}, and map it through $\widehat{\Delta}$ to $\widehat{R_E}$ in the expert's space. Since any point we map to must strictly induce the expert's policy (with slack at least ϵ), we can think of the space we map

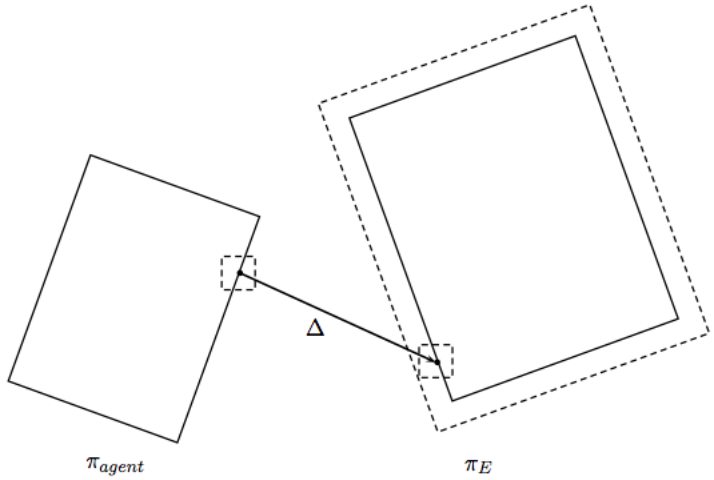


Figure 3.7: Visualization of Lemma 4.

to as the interior (shown with solid lines) of the expert's reward space, where the dotted boundary represents expert rewards that do not strictly induce the expert's policy. For any points on and within the solid lines, we are guaranteed to have at least an open hypercube around the point that strictly induces the expert's policy. When $R + \Delta$ induces a non-expert policy, we know that the open hypercube of points centered at $\widehat{R}$ that maps to the hypercube centered $\widehat{R_E}$ cannot be the agent's actual reward.

In addition to this hypercube of points that maps to the expert's policy, any reward R such that $(R + \widehat{\Delta}) \in \text{IRL}^{\widehat{\pi}}_{strict}$ for any policy $\widehat{\pi} \neq \pi'$ must not be the agent's true reward. However, Lemma 4 and these observations do not directly lead to the claim that Algorithm 1 completes in a finite number of iterations. First, as shown in Figure 3.7, some points with the bounded volume around $\widehat{R}$ may not be in the IRL space of the agent's policy; the amount of space cut by the added constraint may only be a fraction of the volume of the hypercube. Second, the volumes bounded by different constraints may intersect, again suggesting that an additional constraint may only cut a fraction of the volume of the hypercube from the space of possible rewards.

One way to deal with these issues is to try to find a bound on the volume of space eliminated from the space of agent rewards at every iteration. Given such a bound, since the space of agent rewards is bounded, it must be the case that we can cover the entire space within a finite number of iterations. However, the volume eliminated cannot be bounded:

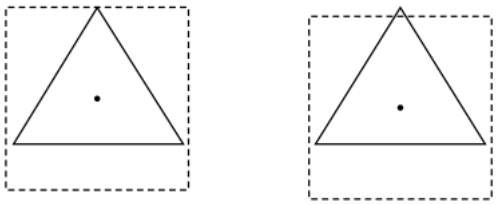


Figure 3.8: Construction from Claim 2

Claim 2. The volume of space eliminated by an added constraint can be arbitrarily small.

Proof. We show this by a simple 2D construction as depicted in Figure 3.8. Consider the space of possible agent rewards as a triangle inscribed within an open square (that is, the vertices of the triangle are barely in the interior of the square). The open square has side length $s = 2\delta$, where 2δ is the side length of the hypercube around any point eliminated. If the point guessed as the agent's reward is at the center of the square, clearly the entire space is covered. However, if the point guessed were slightly off center, a small volume of the triangle would be left uncovered. Since this volume can be arbitrarily small, the next point guessed will only remove an arbitrarily small volume from the space of rewards. □

In the construction shown in Figure 3.8, even though the next point can only cover an arbitrary volume, it is enough to cover the rest of the space. While we cannot bound the volume eliminated by each added point, we may still be able to bound the number of points that can be guessed before the entire space is covered.

By a pigeonhole argument, we show that only a finite number of such points can fit within the space:

Lemma 5. *Given a bounded subspace K in $\mathbb{R}^n$, for some $r > 0$, we place a set of points $S = \{x, y : |x_i - y_i| > r, 1 \leq i \leq n\}$ in K. S must be finite.*

Proof. We start by partitioning K into a grid of n-dimensional cubes (see Figure 3.9 for a 2D depiction) each with side length r, such that the farthest distance along any one dimension between any two points within a cube is r. Since K is bounded, it must be covered by some finite number m of such cubes. Consider a point being placed in K; it must fall within one of the cubes. Since there is no other point within the cube that is more than r away along any dimension, no other points can be placed within the cube where the point resides. Since each point must take up at least one cube and there is a finite number of cubes that cover K, only a finite number of points can be placed in K. □

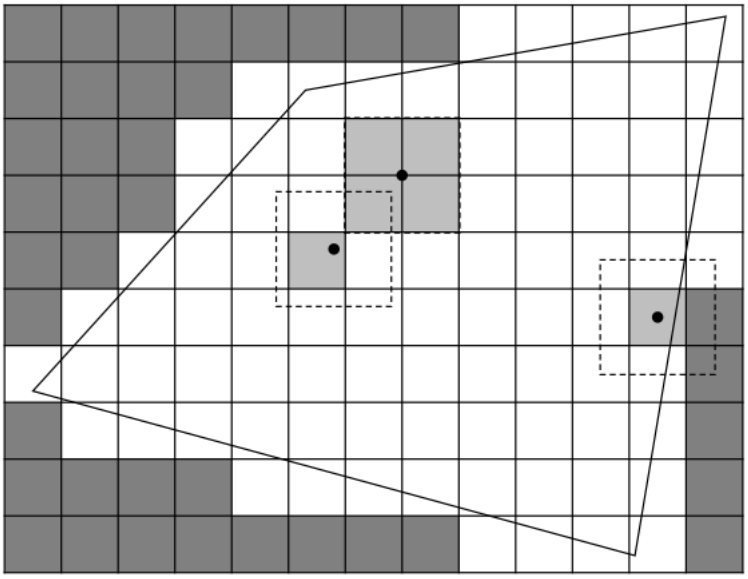


Figure 3.9: Visualization of Lemma 5.

Theorem 3. *Algorithm 1 terminates in a finite number of steps with an admissible mapping Δ, or returns FAILURE if no such mapping exists.*

Proof. Every iteration of Algorithm 1 finds $\widehat{R}$ and $\widehat{\Delta}$ that strictly induce the expert's policy π_E with slack at least ϵ. By Lemma 4, each $\widehat{R}$ eliminated has an open hypercube with positive side length $2\delta = \frac{\epsilon(1-\gamma)}{\gamma}$ centered at $\widehat{R}$ such that all points within this hypercube are not the agent's reward function. Since points in this hypercube are not considered in future iterations, no $\widehat{R}$ may be within more than one hypercube. This implies that any $\widehat{R}$ to be eliminated must be at least δ away from any other $\widehat{R}$ eliminated. Since $R(s)$ is bounded (by Assumption 2) and the state space is finite (by Assumption 1), Lemma 5 ensures that only a finite number of points $\widehat{R}$ may be placed in the space of agent rewards such that each point is more than $(1-\kappa)\delta$ away from another, for some small $\kappa > 0$. This implies that only a finite number of guesses $\widehat{R}$ need to be eliminated in order to cover the entire space of possible agent rewards. Algorithm 1 will terminate in a finite number of steps either by finding an admissible mapping Δ before the entire space is covered, or returning FAILURE once the space is all covered. □

This proof makes use of the minimal slack ϵ that is guaranteed when choosing a reward function in the IRL space of the expert's policy. To visualize this, we can think of the expert's reward space that is being mapped to as having a border with a width of δ as shown in Figure 3.7. As such, every mapped-to point has an open hypercube around it such that all points in this open hypercube still induce the expert's policy. This implies that there is an open hypercube of the same size around a possible agent reward $\widehat{R}$, such that through $\widehat{\Delta}$, there is a one-to-one mapping from each of these points to the points in the hypercube around the expert's reward, points that all strictly induce the expert's policy. As such, these points violate the constraint that $(R + \widehat{\Delta}) \in IRL^{\pi'}$ and are removed from consideration. Figure 3.9 shows that each point removed this way must completely cover at least one new square (shown in light gray) from the grid of squares that covers the agent's reward space, even though the spaces covered by the squares may intersect or be outside the space of agent rewards. This ensures that we can cover the space in a

finite number of iterations.

The number of iterations necessary to complete the elicitation process is dependent on the minimal slack ϵ and the discount factor γ. Increasing the slack decreases the number of potential iterations required to cover the entire space. Generally, as long as ϵ is set to be much smaller than the limits on the incentives provided, it is virtually impossible for ϵ to prevent an admissible mapping to the expert's reward space that would have otherwise existed were ϵ any smaller.

3.2.4 Domains without an Expert

We have provided a general elicitation algorithm for policy teaching in domains with an expert. In trying to come up with an elicitation algorithm for domains without an expert, we are confronted with a couple of new issues. First, in domains with an expert, there is the expert's policy to map to such that the elicitation process completes as soon as we hit any points in the reward space of that policy. In domains without an expert, we do not have a specific policy to map to. We can choose an optimal policy in the interested party's problem as a target, but mapping to such a policy is unlikely to be feasible.

Second, in domains without an expert with known rewards, we can solve an optimization problem to find an admissible mapping based on the agent's reward R to a target policy π_T that maximizes the value of the start state of the interested party's problem. With unknown rewards, we do not know R, and thus do not know π_T. We can guess an agent reward $\widehat{R}$ and solve a similar optimization problem to find a mapping $\widehat{\Delta}$ to a policy $\widehat{\pi_T}$ with respect to $\widehat{R}$. However, even if the agent's motivated reward $R + \widehat{\Delta}$ induces $\widehat{\pi_T}$, the value of $\widehat{\pi_T}$ may be lower than the value of π_T with respect to the interested party's problem. Furthermore, even when $\widehat{\pi_T} = pi_T$, the elicitation method may not know that this is the best achievable policy and continue to look for mappings to policies with higher value. We do not have this problem in domains with an expert, where we are always attempting

to map to the same policy π_E.

Finally, what happens if the agent does perform $\widehat{\pi_T}$? While it is clear that we cannot declare our elicitation process as complete, what constraint can we add to cut the space of agent rewards? Without contrary evidence to our expectation, adding an IRL constraint $R + \widehat{\Delta} \in \text{IRL}^{\widehat{\pi_T}}$ does not remove $\widehat{R}$ from the space of agent rewards.[1] To ensure that we make progress at every iteration of the elicitation process, we must come up with some other constraint to remove points from consideration.

We address these issues in turn. First, the goal of policy teaching without an expert should still be to maximize the value in the start state of the interested party's problem. This must be done with respect to the *unknown* agent reward R:

Definition 17. Policy teaching in domains without an expert
An agent faces a MDP $M = \{S, A, R, P, \gamma\}$ and performs the optimal policy π. An interested party knows the problem definition MDP\R $M_{-R} = \{S, A, P, \gamma\}$, observes the agent's policy, but does not know the agent's reward R. The interested party faces a MDP T = $M_{-R} \bigcup \{G\}$, and may provide an admissible incentive Δ to modify the agent's reward function to $R' = R + \Delta$. Can the interested party learn about the agent's reward function as to be able to provide Δ such that the agent's modified policy π' maximizes the value in the start state of the interested party's problem with respect to the $unknown$ agent reward R?

The difficulty in solving this problem is that we don't know the best achievable mapping with respect to the unknown reward R. However, given a reward guess $\widehat{R}$ and a modified agent policy π', we can calculate the value function GV for the interested party's problem with respect to π' and the interested party's reward G by solving the following system of equations:

[1] In fact, the constraint does not remove any points within the hypercube around $\widehat{R}$ that induce $\widehat{\pi_T}$ through mapping $\widehat{\Delta}$.

$$GV^{\pi'}(s) = G(s) + \gamma \sum_{s'} P(s, s', \pi'(s)) GV^{\pi'}(s) \quad \forall s \in S \tag{3.26}$$

We can observe the value $GV^{\pi'}(start)$ in the start state and compare it to the value $GV^{\pi}(start)$ based on the agent's original policy. If $GV^{\pi'}(start) > GV^{\pi}(start)$, we update $\widehat{\Delta}_{best} = \widehat{\Delta}$ that induced π' and $\widehat{GV}_{max} = GV^{\pi'}(start)$ as the best mapping and value found so far. In choosing a guess for the agent's reward, we can restrict our attention to rewards that have an admissible mapping to a policy that would induce $GV(start) > \widehat{GV}_{max}$. This can be accomplished using the following constraints (on variables R and Δ):

$$Q(s, a_{si}) = R(s) + \Delta(s) + \gamma \sum_{s'} P(s, s', a_{si}) V(s') \qquad \forall s, i \tag{3.27}$$

$$V(s) \geq Q(s, a_{si}) \qquad \forall s, i \tag{3.28}$$

$$V(s) \leq M_v(1 - X_{si}) + Q(s, a_{si}) \qquad \forall s, i \tag{3.29}$$

$$V(s) - Q(s, a_{si}) + \epsilon X_{si} \geq \epsilon \qquad \forall s, i \tag{3.30}$$

$$GQ(s, a_{si}) = G(s) + \gamma \sum_{s'} P(s, s', a_{si}) GV(s') \qquad \forall s, i \tag{3.31}$$

$$GV(s) \geq -M_{gv}(1 - X_{si}) + GQ(s, a_{si}) \qquad \forall s, i \tag{3.32}$$

$$GV(s) \leq M_{gv}(1 - X_{si}) + GQ(s, a_{si}) \qquad \forall s, i \tag{3.33}$$

$$GV(start) \geq \widehat{GV}_{max} + \kappa \tag{3.34}$$

$$\sum_{s} \Delta(s) \leq D_{max} \tag{3.35}$$

$$\Delta(s) \geq 0 \qquad \forall s \tag{3.36}$$

$$\sum_{i} X_{si} = 1 \qquad \forall s \tag{3.37}$$

$$X_{si} \in \{0, 1\} \qquad \forall s, i \tag{3.38}$$

where M_v and M_{gv} are set as in Program 2.8 and ϵ and κ are small constants > 0. Constraint 3.30 ensures that the target policy based on $R + \Delta$ is strictly optimal for $\widehat{R}$ and $\widehat{\Delta(s)}$ found this way.[2] Constraint 3.34 limits the attention to only agent

[2]This constraint works the same way as the strict IRL constraints. Here the constraint ensures that the Q value of actions not taken ($X_{si} = 0$) are at least ϵ less than that of the chosen action.

rewards that have an admissible Δ to a policy with $GV(start) > GV_{max}$. We denote the space of such rewards as $R \in R_{>gv_{max}}$. The other constraints are identical to the ones in Program 2.8 used for policy teaching with known rewards.

If we cannot find a reward $\widehat{R}$ that simultaneously satisfies IRL constraints and is in the space $R_{>gv_{max}}$, we know that there are no mappings from any possible agent rewards that can induce $GV(start) \geq \widehat{GV}_{max} + \kappa$. At this point, we know that $\widehat{\Delta}_{best}$ either solves the policy teaching problem or induces a policy within κ of the best achievable mapping. By setting $\kappa \ll |\widehat{GV}_{max}|$, we can consider $\widehat{\Delta}_{best}$ and $\widehat{GV}_{max}$ close enough to optimal and end the elicitation process.

Based on agent reward guess $\widehat{R}$ selected in this manner, we wish to find $\widehat{\Delta}$ that gives the highest $GV(start)$ with respect to $\widehat{R}$:

$$\max_{\Delta} \; GV(s_{start}) \tag{3.39}$$

subject to:

$$Q(s, a_{si}) = \widehat{R}(s) + \Delta(s) + \gamma \sum_{s'} P(s, s', a_{si}) V(s') \qquad \forall s, i \tag{3.40}$$

$$V(s) \geq Q(s, a_{si}) \qquad \forall s, i \tag{3.41}$$

$$V(s) \leq M_v(1 - X_{si}) + Q(s, a_{si}) \qquad \forall s, i \tag{3.42}$$

$$V(s) - Q(s, a_{si}) + \epsilon X_{si} \geq \epsilon \qquad \forall s, i \tag{3.43}$$

$$GQ(s, a_{si}) = G(s) + \gamma \sum_{s'} P(s, s', a_{si}) GV(s') \qquad \forall s, i \tag{3.44}$$

$$GV(s) \geq -M_{gv}(1 - X_{si}) + GQ(s, a_{si}) \qquad \forall s, i \tag{3.45}$$

$$GV(s) \leq M_{gv}(1 - X_{si}) + GQ(s, a_{si}) \qquad \forall s, i \tag{3.46}$$

$$\sum_{s} \Delta(s) \leq D_{max} \tag{3.47}$$

$$\Delta(s) \geq 0 \qquad \forall s \tag{3.48}$$

$$\sum_{i} X_{si} = 1 \qquad \forall s \tag{3.49}$$

$$X_{si} \in \{0, 1\} \qquad \forall s, i \tag{3.50}$$

This is the same as Program 2.8 with strictness constraints added on the target policy. We keep Program 3.39 separate from the process of finding $\widehat{R}$ so that any objective may be used in picking a $\widehat{R}$.[3] Based on $\widehat{\Delta}$ found using Program 3.39, we expect the agent's motivated policy to be the targeted policy $\widehat{\pi_T}$ if our guess $\widehat{R}$ were correct. If the agent performs any other policy π', we can eliminate $\widehat{R}$ from the space of possible agent rewards by an additional IRL constraint $(R + \widehat{\Delta}) \in IRL^{\pi'}$. If the agent performs $\widehat{\pi_T}$, we can update $\widehat{GV}_{max} = GV^{\pi'}$(start) and restrict our attention to rewards that can map to policies with higher $GV(start)$. Since the best achievable $GV(start)$ based on $\widehat{R}$ is $GV^{\pi'}$, $\widehat{R}$ must be eliminated from consideration. This guarantees that we are making progress at each iteration of the elicitation process. We have the following algorithm:

Algorithm 2 Eliciation Method in domains without an expert

Given policy π, interested party reward G; variables R, Δ; constraint set $K = \emptyset$
Add $R \in$ IRL$^{\pi}$ to K
$\widehat{GV}_{max}$ = $GV^{\pi}(start)$, $\widehat{\Delta}_{best}$ = 0.
Add $R \in R_{>gv_{max}}$ to K
Add $admissible(\Delta)$ to K
loop
  Find values $\widehat{\Delta}$, $\widehat{R}$ that satisfies all constraints in K
  if no such Δ exists **then**
    return $\widehat{\Delta}_{best}$
  else
    Find $\widehat{\Delta}$ and expected policy $\widehat{\pi_T}$ based on $\widehat{R}$ and Program 3.39
    Provide agent with incentive $\widehat{\Delta}$
    Observe agent policy π' with respect to $R' = R^{true} + \widehat{\Delta}$.
    if $GV^{\pi'}(start) > \widehat{GV}_{max}$ **then**
      $\widehat{GV}_{max}$ = $GV^{\pi'}(start)$, $\widehat{\Delta}_{best}$ = $\widehat{\Delta}$.
      Modify $R \in R_{>gv_{max}}$ in K to reflect change in $\widehat{GV}_{max}$.
    end if
    Add $(R + \widehat{\Delta}) \in$ IRL$^{\pi'}$ to K
  end if
end loop

[3]Were we to use one program, we would always find $\widehat{R}$ and $\widehat{\Delta}$ with the highest $GV(start)$ out of all possible rewards. This is a possible heuristic for picking $\widehat{R}$, but the actual agent reward may not have a mapping to a policy with such a high value.

Algorithm 2 formalizes the elicitation process for domains without an expert. At every iteration, we add an IRL constraint based on π' whether $\pi' = \widehat{\pi_T}$ or not; this ensures that we incorporate any evidence gathered even if a generated constraint is not guaranteed to eliminate points from the space. We make the same kind of completion guarantee as we make with Algorithm 1:

Theorem 4. *Algorithm 2 terminates in a finite number of steps with an admissible Δ that maps to a policy whose GV(start) is within κ of the maximium GV(start) achievable with respect to the agent's true reward.*

Proof. Every iteration of Algorithm 2 finds $\widehat{R}$ and $\widehat{\Delta}$ that strictly induces a policy $\widehat{\pi_T}$ with slack of at least ϵ. If the agent performs a policy $\pi' \neq \widehat{\pi_T}$, the added IRL constraint $R + \widehat{\Delta} \in \text{IRL}^{\pi'}$ ensures that $\widehat{R}$ and all points within an open hypercube with side length $2\delta = \frac{\epsilon(1-\gamma)}{\gamma}$ centered at $\widehat{R}$ are not the agent's reward function. If $\pi' = \widehat{\pi_T}$, $\widehat{GV}_{max}$ increases. Since Program 3.39 ensured that $\widehat{\Delta}$ mapped $\widehat{R}$ to its highest achievable $GV(start)$, the constraints $R_{>gv_{max}}$ ensure that $\widehat{R}$ and all points within an open hypercube with side length $2\delta = \frac{\epsilon(1-\gamma)}{\gamma}$ centered at $\widehat{R}$ are not the agent's reward function. It follows by an application of Lemma 5 that only a finite number of $\widehat{R}$ need to be eliminated in order to cover the space of possible agent rewards. Since Algorithm 2 only terminates when there is no possible agent reward that can achieve a $GV(start)$ of at least $\widehat{GV}_{max} + \kappa$, the returned admissible $\widehat{\Delta}$ maps the agent's true reward to a policy with $GV(start)$ within κ of the maximum achievable $GV(start)$ with respect to the agent's true reward. □

3.2.5 Elicitation Objective Function

Algorithms 1 and 2 allow us to solve the policy teaching problem with unknown rewards using general elicitation methods that are guaranteed to terminate within a finite number of steps. The methods allow for any objective function to be used for choosing $\widehat{R}$ and $\widehat{\Delta}$ that satisfy the constraints on R and Δ. In choosing an objective function, we are interested in the following properties:

- **Few elicitation rounds in practice**

  While we have guarantees on the maximum number of rounds that the elicitation process may take, we wish to complete the elicitation process in as few rounds as possible. In domains without an expert, we need to solve computationally expensive Mixed Integer Programs at every iteration. In both domains with and without an expert, we wish to limit the amount of interaction we need to have with the agent before finding a solution.

- **Robustness**

  In cases where elicitation is lengthy or costly, we may wish to perform only a couple of rounds of elicitation and return the best mapping found before the policy teaching problem is solved. In these situations, choosing potential mappings that can guarantee a certain level of GV(start) with respect to uncertainty over the agent's rewards may be more important than reducing the number of elicitation rounds.

- **Tractability**

  Objective functions that satisfy the above properties must be computationally tractable to be applicable in practice.

These properties are all desirable, but any particular objective function is unlikely to achieve all three. In some cases, there may be heuristics that do not make guarantees but perform well in practice. We will discuss the minimax regret criterion for guarantees on the quality of induced policies and methods of cutting the reward space to reduce the number of elicitation rounds.

Minimax Regret

The minimax regret decision criterion provides a method of making robust decisions. The criterion bounds the worst case error in making a decision given uncertainty over preferences. The criterion does not require a prior over possible agent

rewards, which is consistent with our setup.[4] In adopting this criterion, we follow the formulation used by Wang and Boutilier [25]. In the policy teaching problem, we aim to choose $\widehat{R}$ and $\widehat{\Delta}$ that bounds the worst case GV(start) of the induced agent policy based on $R_{true} + \widehat{\Delta}$ given uncertainty over R.

Definition 18. A *decision* a is a pair $\langle \widehat{R}, \widehat{\Delta} \rangle$ that satisfies all constraints on R and Δ.

Definition 19. The *pairwise regret* of making decision a over decision a' is the worst case difference in GV(start) of induced policies with respect to possible agent rewards u:

$$R(a, a') = \max_{u} GV^{\pi'}_{start}(a', u) - GV^{\pi}_{start}(a, u) \tag{3.51}$$

where π' is the policy induced by u and incentives based on a' and π is induced by u and incentives based on a. u is subject to the same constraints on $\widehat{R}$ chosen from the space of possible rewards.

Definition 20. The *maximum regret* of choosing a decision a is:

$$MR(a) = \max_{a'} R(a, a') \tag{3.52}$$

Definition 21. The *Minimax regret decision criterion* chooses a decision that minimizes the max regret. The *minimax regret* is:

$$MRR = \min_{a} MR(a) \tag{3.53}$$

The minimax regret decision criterion makes robust decisions in choosing $\widehat{R}$ and $\widehat{\Delta}$. At every iteration, new evidence on R shrinks the space of possible rewards; this guarantees that the minimax regret can only decrease or stay the same. If the minimax regret becomes low enough, the interested party may be satisfied with the mapping found and decide to quit before finding the best achievable mapping. [5]

[4] Regret based methods have also been shown to be effective even in the presence of priors [25], and could still be useful were we to have priors in our domains

[5] In policy teaching without an expert, Algorithm 2 can be seen as terminating the elicitation process when the maximum regret in choosing the best mapping found so far over any other mappings is minimal with respect to possible agent rewards.

While the minimax regret decision criterion provides robustness, it does not lead to fewer rounds of elicitation before coming up with a good mapping. The concern over worst-case error can lead the elicitation process away from mappings that would actually perform well with respect to the agent's true reward R. For example, a decision that performs well with respect to most rewards in the space of possible rewards will not be taken if there are *any* other rewards in the space (which are not even the agent's actual reward) for which the decision does not perform well. This conservative approach provides robust guarantees, but may require many rounds of elicitation before finding a good mapping.

More problematically, the computation of the minimax regret decision is intractable. While some techniques have been developed for some domains with factored preferences [6], these techniques do not apply to our domain. For the policy teaching problem, even the computation of pairwise regret requires coming up with a policy based on a guessed reward and incentives provided and evaluating the policy with respect to the interested party's value function, which is itself a computationally expensive process.

Information Gain

While the minimax regret criterion provides a robust method for making decisions, the method is intractable for our domain and does not lead to fewer rounds of elicitation before coming up with a good mapping. In practice, we are more interested in an objective that can lead to quicker progress in the elicitation process. One possible approach is to try to come up an objective that makes decisions with a high likelihood of finding good mappings. However, without any prior information on the space of possible agent rewards, it is unclear how to form such an objective that works well in general. Furthermore, even if we can find a good mapping quickly, the interested party may be interested in the optimal mapping; it may still take many rounds of elicitation to completely solve the policy teaching problem.

A different approach is to try to come up with an objective that chooses $\widehat{R}$ and $\widehat{\Delta}$ that maximize the information gain about the agent's true reward. If we can eliminate a large portion of the space of possible agent rewards at every iteration of the elicitation process, we can solve the policy teaching problem in fewer iterations. Unfortunately, coming up with the optimal decision that maximizes the expected value of information is an intractable problem [5].

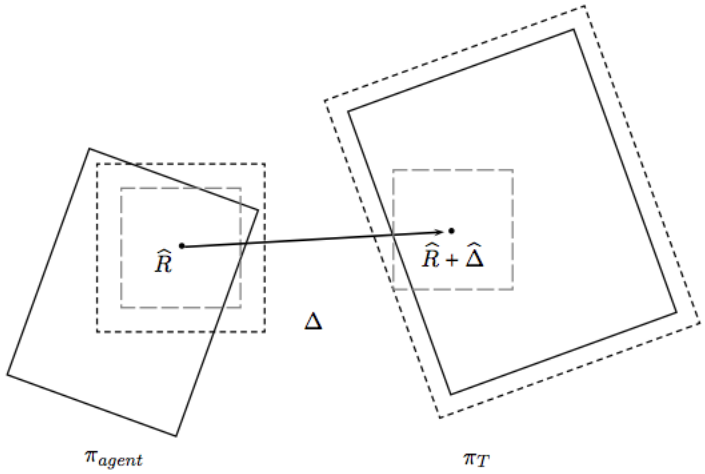


Figure 3.10: Choosing $\widehat{R}$ and $\widehat{\Delta}$ to maximize the slack in both the agent's reward space and the target space.

Nevertheless, we can heuristically choose $\widehat{R}$ and $\widehat{\Delta}$ that has a high likelihood of significantly narrowing the space of possible agent rewards. From Lemma 4, we know that the amount of space that can be eliminated from the space of agent rewards is based on the slack in the Q values of the policy induced by $\widehat{R}+\widehat{\Delta}$. If we attempt to map to a point with a high slack, we will have a large volume of points around $\widehat{R}$ that we know are not the agent's true reward. Furthermore, if a large volume of these points lie within the current space of possible agent rewards, we are guaranteed to make a large cut to the space. Figure 3.10 shows points being picked that have a lot of slack in both the agent's reward space and the target space. The inner square around $\widehat{R}$ denotes the space of points that can be eliminated based on added constraints.

In coming up with an objective, we aim to maximize both the slack on the agent's policy based on $\widehat{R}$ and the slack around the target policy based on $\widehat{R} + \widehat{\Delta}$. In domains with an expert, the target policy is the expert's policy π_E, and the slack is based on a reward $R_E \in IRL^{\pi_E}_{strict}$. We can write down the objective and the associated constraints with a simple linear program:

$$\max_{\Delta}[\sum_s \beta(s) + \sum_s \beta_E(s) - \lambda \sum_s \alpha(s)] \tag{3.54}$$

$$\alpha(s) \geq R(s) \qquad \forall s \in S \tag{3.55}$$

$$\alpha(s) \geq -R(s) \qquad \forall s \in S \tag{3.56}$$

$$((\mathbf{P}_\pi - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-\mathbf{1}}\mathbf{R})[s] \geq \beta(s) \qquad \forall a \in A, s \in S \tag{3.57}$$

$$((\mathbf{P}_{\pi_\mathbf{E}} - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P}_{\pi_\mathbf{E}})^{-\mathbf{1}}\mathbf{R_E})[s] \geq \beta_E(s) \qquad \forall a \in A, s \in S \tag{3.58}$$

$$(\mathbf{P}_\pi - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-\mathbf{1}}\mathbf{R} \succeq \mathbf{0} \qquad \forall a \in A \tag{3.59}$$

$$(\mathbf{P}_{\pi_\mathbf{E}} - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P}_{\pi_\mathbf{E}})^{-\mathbf{1}}\mathbf{R_E} \succeq \epsilon \qquad \forall a \in A \tag{3.60}$$

$$\sum_s \Delta(s) \leq D_{max} \tag{3.61}$$

$$\Delta(s) \geq 0 \qquad \forall s \tag{3.62}$$

where β is the slack in the agent's Q values, β_E is the slack in the expert's Q values, and α is the absolute value of the picked agent reward. The λ constant and the α variables place a weighted penalty on large rewards, which allows us to prefer simpler rewards and also to avoid picking large rewards for the sake of increasing the slack.[6] While this program does not provide any guarantees like minimax regret, it is a tractable linear program that is likely to significantly reduce the number of iterations in the elicitation process.

We can similarly attempt to maximize the slack for problems in domains without an expert. In these domains, $\widehat{R}$ and $\widehat{\Delta}$ are chosen separately; an optimization problem that simultaneously finds a $\langle \widehat{R}, \widehat{\Delta} \rangle$ pair with the highest stack and ensures

[6] If we wish, we can place penalty terms on the expert's reward as well. We can also place additional constraints to weigh or balance the values of β and β_E.

that $\widehat{\Delta}$ maps $\widehat{R}$ to a policy with the highest achievable GV(start) is computationally intractable. However, we can achieve much of the same effect by using one of the following tractable alternatives:

1. Choose $\widehat{R}$ with maximal slack for the agent's policy. Find highest reachable $\widehat{GV}_{max}$ based on $\widehat{R}$. Find $\widehat{\Delta}$ with maximal slack such that $\widehat{R} + \widehat{\Delta}$ has $GV(start) = \widehat{GV}_{max}$.

2. Find highest achievable $\widehat{GV}_{max}$ based on any $\widehat{R}$ and $\widehat{\Delta}$. Find $\widehat{R}$ and $\widehat{\Delta}$ with maximal slack such that $\widehat{R} + \widehat{\Delta}$ has $GV(start) = \widehat{GV}_{max}$.

3. Find $\widehat{R}$ and $\widehat{\Delta}$ with maximal slack without requiring $\widehat{\Delta}$ to map $\widehat{R}$ to the best achievable policy. Based on $\widehat{R}$, find a $\widehat{\Delta}$ with maximal slack such that $\widehat{R} + \widehat{\Delta}$ has GV(start) $= \widehat{GV}_{max}$.

The first alternative provides a greedy method by finding $\widehat{R}$ with maximal slack and then finding Δ with maximal slack based on $\widehat{R}$. The second alternative restricts attention to $\widehat{R}$ that can induce a policy with the highest GV(start) and maximizes the slack within this subset of $\widehat{R}$'s. The third alternative chooses $\widehat{R}$ hoping that the $\widehat{\Delta}$ that provides maximal slack is one that induces the policy with the highest GV(start), and then picks $\widehat{\Delta}$ with maximal slack based on this $\widehat{R}$.

We will adopt the first alternative here because it is less restrictive than the second alternative and more robust than the third. The first alternative is also likely to perform well because it picks $\widehat{R}$ with a large slack such that points around $\widehat{R}$ to be eliminated are still within the space of possible agent rewards. This guarantees that as much progress is made as points eliminated by added constraints.

In choosing $\widehat{R}$, we ensure $R \in IRL^{\pi}$, $R \in R_{>gv_{max}}$, and add additional IRL constraints based on observed induced policies. We use the following objective, with the same β and α constraints in domains with an expert to capture the slack in Q values and to penalize large rewards, respectively:

$$\max_{R,\Delta} \sum_s \beta(s) - \lambda \sum_s \alpha(s) \tag{3.63}$$

$$\alpha(s) \geq R(s) \qquad \forall s \in S \tag{3.64}$$

$$\alpha(s) \geq -R(s) \qquad \forall s \in S \tag{3.65}$$

$$((\mathbf{P}_\pi - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-\mathbf{1}}\mathbf{R})[s] \geq \beta(s) \qquad \forall a \in A, s \in S \tag{3.66}$$

Based on $\widehat{R}$ found this way, we use Program 3.39 to find the maximal achievable $\widehat{GV}_{max}$. We can then find the $\widehat{\Delta}$ with maximal slack that achieves $\widehat{GV}_{max}$:

$$\max_{\Delta} \sum_s \beta(s) \tag{3.67}$$

subject to:

$$Q(s, a_{si}) = \widehat{R}(s) + \Delta(s) + \gamma \sum_{s'} P(s, s', a_{si}) V(s') \qquad \forall s, i \tag{3.68}$$

$$V(s) \geq Q(s, a_{si}) \qquad \forall s, i \tag{3.69}$$

$$V(s) \leq M_v(1 - X_{si}) + Q(s, a_{si}) \qquad \forall s, i \tag{3.70}$$

$$V(s) - Q(s, a_{si}) + \epsilon X_{si} \geq \epsilon \qquad \forall s, i \tag{3.71}$$

$$V(s) - Q(s, a_{si}) + M_v X_{si} \geq \beta(s) \qquad \forall s, i \tag{3.72}$$

$$GQ(s, a_{si}) = G(s) + \gamma \sum_{s'} P(s, s', a_{si}) GV(s') \qquad \forall s, i \tag{3.73}$$

$$GV(s) \geq -M_{gv}(1 - X_{si}) + GQ(s, a_{si}) \qquad \forall s, i \tag{3.74}$$

$$GV(s) \leq M_{gv}(1 - X_{si}) + GQ(s, a_{si}) \qquad \forall s, i \tag{3.75}$$

$$GV(start) \geq \widehat{GV}_{max} \tag{3.76}$$

$$\sum_s \Delta(s) \leq D_{max} \tag{3.77}$$

$$\Delta(s) \geq 0 \qquad \forall s \tag{3.78}$$

$$\sum_i X_{si} = 1 \qquad \forall s \tag{3.79}$$

$$X_{si} \in \{0, 1\} \qquad \forall s, i \tag{3.80}$$

where the constants are set as in Program 3.39. Constraint 3.72 ensures that β captures the minimal slack between the optimal action taken and all other actions, and Constraint 3.76 ensures that we find solutions with GV(start) at least $\widehat{GV}_{max}$.

The slack maximizing heuristics we have proposed for domains with and without an expert do not introduce additional computational complexity, but have a high potential for decreasing the number of rounds of elicitation necessary before finding a solution. But how well do these heuristics work in practice? To use these heuristics and our elicitation algorithms, also we need to set the minimal slack ϵ and the reward penalty λ. How should we set these values, and what effects do they have on the length of the elicitation process?

Chapter 4

Experiments

4.1 Experiments

4.1.1 Experimental Setup

We implemented Algorithm 1 and Algorithm 2 with the slack maximizing heuristics introduced in Subsection 3.2.5 to evaluate the performance of our algorithms on our motivating example. We also implemented methods for policy teaching with known rewards as a way of determining whether any feasible mappings exists based on true rewards and the limits on Δ. We used CPLEX version 10.1 to solve the linear programs and mixed integer programs in our algorithms. Our code is written in Java, and uses JOpt[1] as a Java inteface to CPLEX. Experiments are conducted on a local machine with a 2.4Ghz processor and 512MB of RAM.

4.1.2 Setting Parameters

The policy teaching algorithms and the heuristics we use require us to set a couple of parameters. In setting the ϵ for the minimal slack in Q values of policies mapped to, we aim to set the largest value for which we are almost certain we can find a mapping if one exists. To do this, we consider the largest possible increase in Δ that needs to be added for there to be a mapping with a slack of ϵ given that there

[1] http://econcs.eecs.harvard.edu/jopt

exists an admissible mapping. The slack corresponds to a reward pertubation of at most $\frac{\epsilon(1-\gamma)}{2\gamma}$ in each state; for a domain with n states, the maximal amount of Δ we may need to have to add is $\delta^+ = \frac{n\epsilon(1-\gamma)}{2\gamma}$. If we can set ϵ such that δ^+ is a small percentage of D_{max} (i.e. 1%), we are almost certain that there will be a mapping with slack ϵ if there is an admissible mapping.[2]

For our slack maximizing objective functions, we need to set the reward penalizing coefficient λ. As we have briefly mentioned, penalty terms on the reward function allow the objective to prefer "simpler" rewards and avoid choosing large rewards for the sake of increasing the slack. Simple rewards provide a simple explanation for the agent's behavior. Without any penalty terms, the objective is likely to add bumps to the reward function that would increase the slack but do little extra to explain the agent's behavior. While this seems to suggest that we should set λ to be large, we do not want to set λ to be so large as to focus on penalizing rewards over maximizing the slack that our heuristic depends on for increasing the information gain.

Ng and Russell [16] observed that since $\mathbf{R} = \mathbf{0}$ is always a solution to (nonstrict) IRL constraints, for large enough λ, the penalty on rewards would cause $\mathbf{R} = \mathbf{0}$ to be the solution to the slack maximizing objective in Program 3.17.[3] They proposed choosing the "simplest" non-zero $\mathbf{R}$ by setting λ right below the phase transition λ_0, where $\mathbf{R} \neq \mathbf{0}$ for $\lambda < \lambda_0$, and $\mathbf{R} = \mathbf{0}$ for $\lambda > \lambda_0$. Setting λ this way provides a natural way of picking rewards, and avoids having to hand-tune the parameter.

However, λ set this way is unlikely to be a good choice for the policy teaching problem. First, setting λ right below the phase transition produces smooth reward functions with zero rewards in most states. For Ng and Russell's problem domain, this provides a fine choice because they aimed to recover a simple and generalizable reward function. For the policy teaching problem where an agent is likely to

[2] δ^+ is based on a worst case analysis. Setting it at 1% of D_{max} is a conservative estimate.

[3] The objective has value 0 when $R = 0$. Our objectives in our heuristics are slightly different and do not have this property, as we deal with expert policies (with strictness constraints preventing $R_E = 0$) in one objective and GV_{max} constraints for domains without an expert.

have conflicting interests and personal costs and rewards, the agent's true reward is unlikely to be so smooth as to be 0 in most states.

Second, rewards found this way are prone to have small slack in many states. Even if the slack is large in a few states, the total volume that can be bounded would still be low because it is based on the product of the slacks in each of the states. Furthermore, since the phase transition corresponds to a switchover to positive objective values in Program 3.17, the total slack is likely to also be low, as it is just slightly larger than the reward penalty. The amount of information gain we can achieve based on setting λ at the phase transition may be limited.

To find the middle space between maximizing the slack and penalizing large rewards, we set $\lambda = \lambda_0/2$. Without any additional information on the agent's reward, picking λ this way should provide an improvement (or be more robust) than picking $\lambda = 0$ or setting λ based on the phase transition. We find λ_0 by a simple search over λ using Program 3.17 for domains without an expert, and Program 3.54 without the strictness constraints on R_E for domains with an expert.

4.1.3 Testing Example

We perform experiments of our policy teaching algorithms on the child walking home domain, where the child is the agent and the parents are the interested party.

Example 4. Child walking home (for policy teaching)
A child gets off from school and decides to go to the park (and stay there). The child's parents want the child to come home quickly without stopping by the park. The parents observe the child's policy but have no knowledge of the child's reward function. The parents are willing to provide limited incentives to the child to alter his behavior.

We model the problem domain with a MDP $M_{-R} = \{S, A, P, \gamma\}$. Figure 4.1(a) shows the state space S; the child starts at school in the upper left hand corner, and

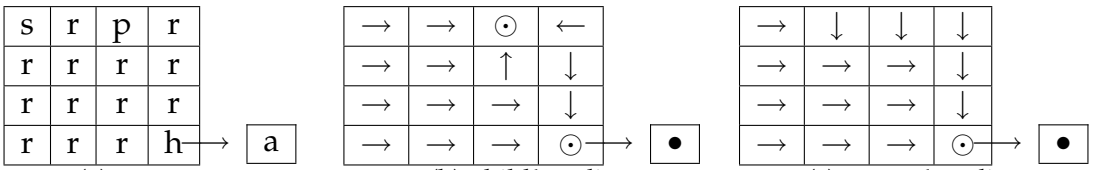


Figure 4.1: Child walking home domain

may choose to walk on the road toward the park on his right, or to walk towards his house at the bottom right corner. The child may move in any horizontal or vertical direction (staying within bounds) or choose to stay in place. The child moves with probability 1 if he chooses a legal direction to move in, or else stays in the state that he is in. The discount factor γ is set to 0.7.

Since incentives are provided on states in our setup, we wish to construct the state space such that the child only receives provided incentives once in the home state that the parents desire him to reach. We accomplish this by introducing an absorbing state, such that once the child reaches home, he will be moved into the absorbing state where he stays forever. The reward and incentives provided in the absorbing state are always set to be 0.

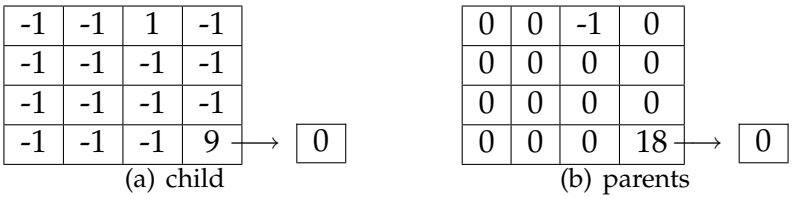


Figure 4.2: Interested party (parents) and agent (child) reward functions

Figure 4.1(b) shows the child's optimal policy with respect to his reward function; he walks to the park from school and stays at the park. We set the child's unobserved reward as shown in Figure 4.2(a). We give the child a -1 reward for being on the road, a +1 reward for the park, and an one time +9 reward for being home.[4] The child does not go home from the park because of the costs for walking

[4]For experiments, the reward and policies can be set any which way (generated or hand-

on the road and the heavy discounting on the reward he receives for being home by the time he reaches there.

We set the parents' reward function as shown in Figure 4.2(b). The parents have a -1 reward for the child being at the park, an one time +18 reward for the child being home, and 0 reward everywhere else. Figure 4.1(c) gives one particular optimal policy with respect to the parents' reward function (any policy that goes directly home without stopping by the park is optimal for the parents' problem). The highest achievable value in the start state is 2.1177.

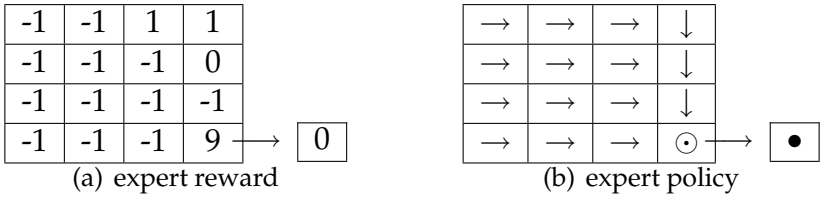
For our experiments, we limit the total amount of incentives provided to $D_{max} = 3$. This setup ensures that there is no way of providing incentives such that the induced child policy is optimal for the parents' problems.[5] The incentives are large enough for there to be a mapping to induce the child to perform a policy that stops by the park but goes home from there. Policies of this kind has value 1.6277 in the start state of the parents problem, and are the only non-optimal policies for the parents' problem with positive value that can be mapped to given $D_{max} = 3$. The incentives are also rather strict; the minimum D_{max} required to induce such a policy is 2.7.

For experiments on policy teaching with an expert, we construct the expert as an older brother whose policy is one which stops by the park but goes home from there. Figure 4.3(a) shows the expert's reward function. He faces the same reward as the agent with extra motivations provided in the top two squares on the rightmost column. We use the expert's optimal policy given by Figure 4.3(b) as the target space to map the child's unknown reward function.

We bound the reward function by R_{max} = 30, which creates a fairly large space from which the algorithms find possible child reward functions. We set $\epsilon = 0.01$ based on $\delta^+ = 16 \times \frac{\epsilon(1-0.7)}{2\times 0.7} = 3.4\epsilon$, such that in the worst case we require about

constructed) as long as we don't use IRL to set the true reward of the child based on his policy.

[5]Verified by methods for policy teaching with known rewards.



(a) expert reward (b) expert policy

Figure 4.3: Expert reward and policy used for experiments

0.034 (about 1% of D_{max}) more incentive to map the child than we would were we not to require strictness. We set $\lambda = 0.78$ based on $\lambda_0 = 1.56$ in domains with an expert, and $\lambda = 0.625$ based on $\lambda_0 = 1.25$ in domains without an expert.

4.2 Results

4.2.1 Domain with an Expert

Our implementation of Algorithm 1 with the slack maximizing heuristic completed in 7 rounds with an admissible Δ mapping the child to the expert's policy shown in 4.3(b). Δ is shown in Figure 4.4(a). The child received incentives to move away from the park to the right and to continue down until he reached home. The provided incentives were enough to offset the -1 cost for walking on the road, as well as provide some positive motivation until the child was close enough to home.

We looked over the $\widehat{\Delta}$ that was provided at each round and the child's induced policy. The process began with the mapping placing a small incentive in the state to the right of the park, with the rest of the incentives being placed below that

0	0	0	1.17
0	0	0	1.78
0	0	0	0.01
0	0	0	0.03

(a) $D_{max} = 3$

0	0	0	1.89
0	0	0	0.81
0	0	0	0.00
0	0	0	0.00

(b) $D_{max} = 2.7$

Figure 4.4: Admissible mappings for domain with an expert. $R_{max} = 30$, $\epsilon = 0.01$, $\lambda = 0.78$.

state. This induced the child to move away from the park, but stay in the state with the larger provided incentive forever without going home. In the subsequent rounds, the proposed mappings placed more and more weight on the state next to the park and less on the state below it. This change led the child to go back to his original policy, since the motivation provided in the state next to the park was not attractive enough and the decreased incentives in the state below were no longer enough to lure the child away from the park. This continued until more incentives were provided in the state next to the park, ending with the admissible mapping shown in Figure 4.4(a).

We then tightened the limits on incentives provided by setting $D_{max} = 2.7$. The elicitation process completed in 19 rounds with the admissible mapping Δ shown in Figure 4.4(b). We see that more incentives have been provided in the state to the right of the park and less as the child moves down the right column. This is exactly what we would expect, as the child is given more immediate (and less discounted) incentives for moving away from the park and just enough motivation in the state below to encourage the child to move towards home. We looked over the $\widehat{\Delta}$ provided at each round, and saw that the first round had most of the incentives in the state below the one to the right of the park, whereas the second round placed most of the incentives in the state to the right of the park. Unsuccessful in both attempts, the mappings in subsequent rounds returned to the earlier observed pattern of shifting incentives towards the state to the right of the park until the admissible mapping was found.

We looked at the rewards that were picked at each round of elicitation, and saw that they were much larger than the actual agent rewards that we had set. This was to be expected, given the way we set λ and the fact that the rewards we set were mostly small rewards. What is interesting about the reward picked is that the relationship between rewards in states were not necessarily off. For example, the reward guessed for the park was 5.67, as compared to 3.64 for the state to the right

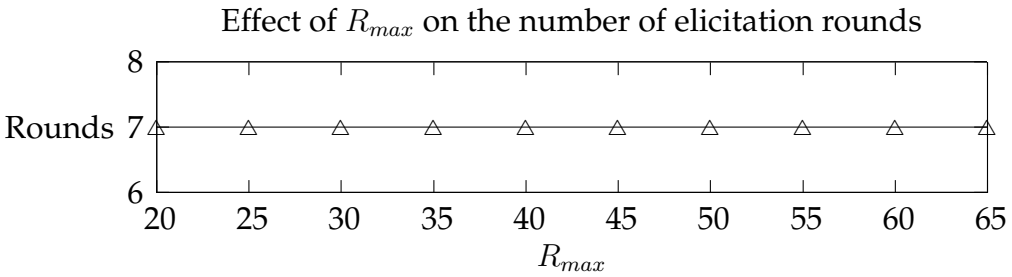


Figure 4.5: Setting R_{max}. $D_{max} = 3$, $\lambda = 0.78$, $\epsilon = 0.01$.

of the park and the one below it. The difference between the rewards is almost exactly the same as the difference in the true rewards we have set, and being able to capture the relationship between states was enough for a mapping to be found in a small number of elicitation rounds.

We continued our experiments by studying the effect of various parameters on the length of the elicitation process. First, we tried to vary the value of R_{max} between a rather tight $R_{max} = 20$ to a very loose $R_{max} = 65$. As we see in Figure 4.5, the length of the elicitation process stayed constant and completed in 7 rounds for each of these cases. This result seems to suggest that our heuristics may be somewhat robust to changes in R_{max}, which is useful since setting R_{max} accurately in practice may be difficult.

Second, we considered the effect of ϵ on the length of the elicitation process. While it is clear that the elicitation process should be shortened as ϵ increases, it is not so clear how quickly and by how much. If we double the value of ϵ, we double the reward perturbation allowed in each state of the reward function. In terms of the hypercube of points that cannot be the agent's rewards, doubling ϵ doubles each side of the hypercube. For a world with 16 states, the volume of each hypercube found this way would increase by a factor of 2^{16}. This suggests that unless our heuristics and added constraints leads to the removal of a much larger volume at each round, we would expect increases in ϵ to have huge effects on the length of the elicitation process.

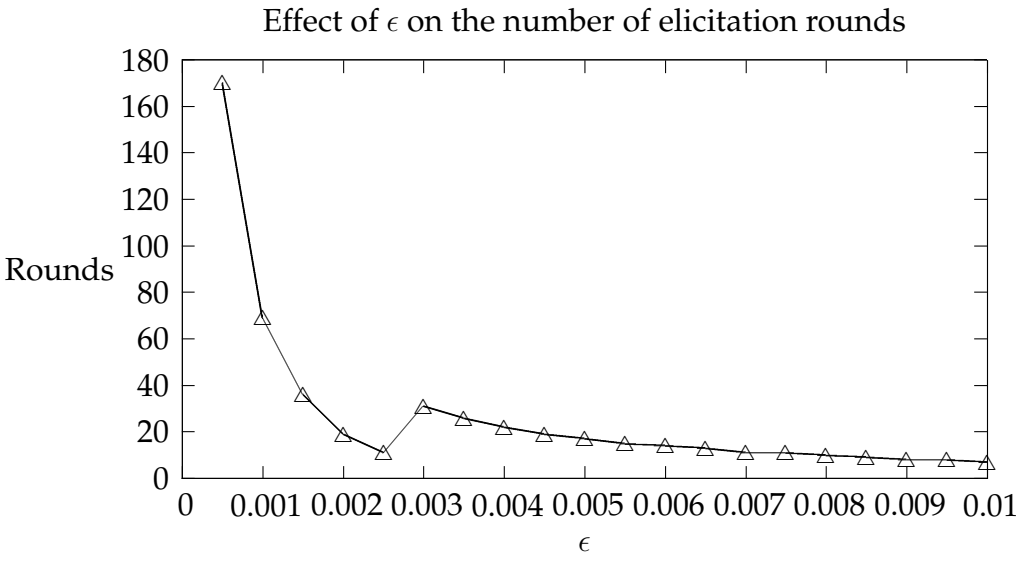


Figure 4.6: Setting ϵ. $D_{max} = 3$, $R_{max} = 30$, $\lambda = 0.78$.

Figure 4.6 shows the effect of varying ϵ from 0.0005 to our set value of 0.01 on the length of the elicitation process. Generally, we see that the number of rounds decreases as ϵ increases as expected, with the number of rounds ranging from 170 to 7. The values on the graph seems to indicate an inversely proportional relationship between ϵ and the number of rounds, such that halving ϵ increases the number of elicitation rounds by a factor of 2 to 3. As compared to a factor of 2^{16} as suggested by the worst case bounds, a factor of 2 to 3 implies that our heuristics and added constraints are doing quite a bit to make large cuts in the space regardless of the minimal slack ϵ. This not only allows the elicitation algorithm to scale well for smaller ϵ values, but also suggests that the algorithms may scale nicely for larger problem domains.

Finally, we study the effect of λ on the length of the elicitation process. We conducted experiments for a large range of λ values that include $\lambda = 0$, our set λ, λ at the phase transition, and λ values greater than the λ_0. We plotted our results on in Figure 4.7, marking our set λ with a darkened triangle and labeling the phase transition. From the left, we see that setting $\lambda = 0$ led to 26 rounds of elicitation,

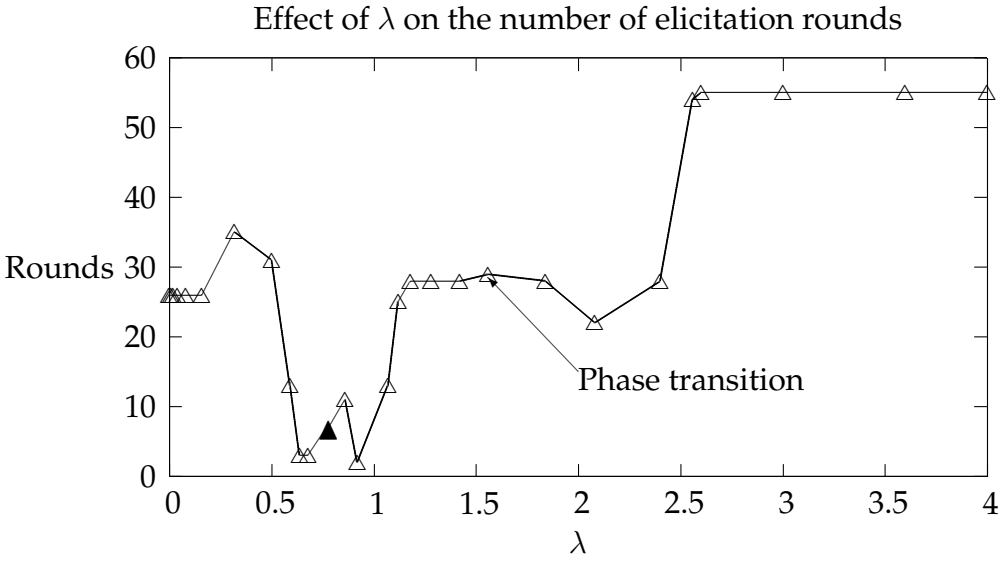


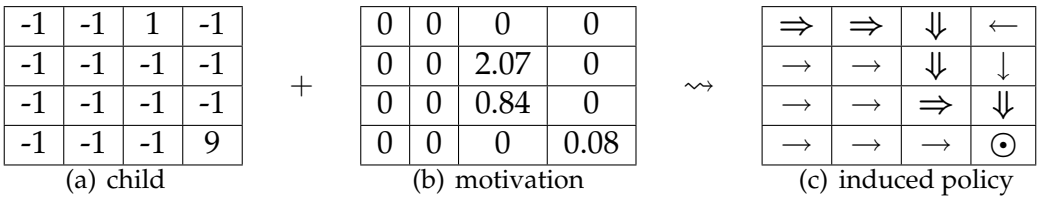
Figure 4.7: Setting λ. $D_{max} = 3$, $R_{max} = 30$, $\epsilon = 0.01$

with the number of rounds increasing slightly followed by a steep drop between $\lambda = 0.5$ and $\lambda = 1$. The number of rounds increased as we moved closer to the phase transition, and continued increasing for larger values of λ.

The larger number of rounds near $\lambda = 0$ and the phase transition in this experiment are in line with our beliefs on why these values may not be good choices for setting λ. λ values greater than the phase transition performed the worst, as these values led to the choosing of uninformative small rewards (in absolute value) with low slack.[6] The lower number of elicitation rounds corresponded to the region between $\lambda = 0$ and the phase transition as we have expected.

We repeated our experiment with $D_{max} = 2.7$ and observed the effect of λ on the number of elicitation rounds. Our results showed that $\lambda = 0$ and other low λ values led to between 31 and 39 rounds of elicitation, with the values dropping to between 18 and 20 rounds for most values up to the phase transition. We observed a similar steep drop in elicitation rounds between λ values of 0.92 and 1.02, where

[6] $\mathbf{R} = \mathbf{0}$ is not always returned as a solution that maximizes the objective because of the strictness constraints placed on the expert's reward. In the first iteration, $\mathbf{R} = \mathbf{0}$ is selected, but the constraint added based on the induced agent policy eliminates $\mathbf{R} = \mathbf{0}$ from being considered again.



(a) child (b) motivation (c) induced policy

Figure 4.8: Domain without an expert. $D_{max} = 3$, $R_{max} = 30$, $\epsilon = 0.01$, $\lambda = 0.625$.

the elicitation process completed in 6 rounds. The λ we set led to 19 rounds of elicitation and was not in this range, but performed better than smaller values of λ and on par with setting λ at the phase transition. We expect setting λ in the way we have described will provide above average results in other domains as well, and acknowledge that there may be other methods for setting λ that can lead to fewer rounds of elicitation.

4.2.2 Domain without an Expert

We ran experiments on our implementation of Algorithm 2 with the slack maximizing heuristic. Using our set parameters, the elicitation process completed in 42 rounds with the mapping Δ shown in Figure 4.8(b). We see that incentives have been provided for states below the park, and these incentives lead the child to move down and away from the park until he reaches home. This induced policy is different than the expert policy we mapped to earlier, but has the same $GV(start)$ value of 1.6277 for the parents' problem.

We examined the $GV(start)$ values of the policies that the algorithm attempted to map the child to at each round of the elicitation process. In the first round, the algorithm successfully aimed and mapped the child to a policy with $GV(start) = 0$, which provided an improvement over the child's policy of staying in the park which had $GV(start) = -1.63$. In all subsequent rounds before the solution was found, the algorithm attempted to map the child to optimal policies for the parents' problem with $GV(start) = 2.117$. Since these policies are unreachable given

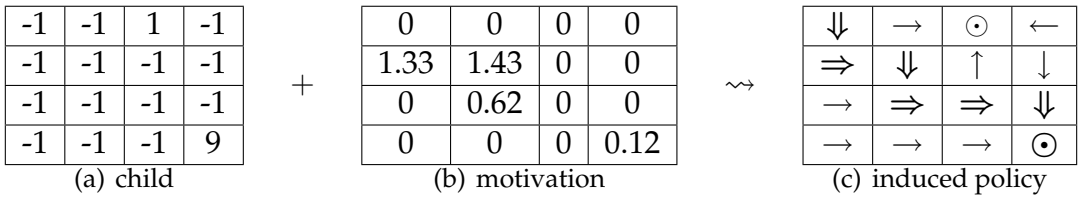
D_{max}	rounds	$\widehat{GV}_{max}$	GV_{max}	GV_{opt}	total time	time/round
3	42	1.6277	1.6277	2.1177	67m 44s	1m 37s

Table 4.1: Domain without an expert. $D_{max} = 3$, $R_{max} = 30$, $\epsilon = 0.01$, $\lambda = 0.625$.

$D_{max} = 3$ and the agent's true reward, the algorithm took iterations to refine its belief about the agent's reward until it found that the best mapping out of the set of possible agent rewards had value $GV(start) = 1.6277$. The algorithm was then able to easily find a mapping to the policy shown in Figure 4.8(c). Given that the algorithm has no way of knowing what the best achievable mapping is and spent most of its time figuring out what mappings are achievable, completing the process in 42 rounds is impressive.

Table 4.1 provides a summary of the process. The $\widehat{GV}_{max}$ found was the best GV_{max} achievable given $D_{max} = 3$. The running time of the algorithm is considerably longer than in domains with an expert, where we averaged about 7 rounds of elicitation per second. Here, each round averaged 1 minute and 37 seconds. While this is reasonable given that the program needs to solve three mixed integer programs per round, this nevertheless shows that the running time may become a bottleneck for larger domains.

We reran the experiment with $D_{max} = 3.5$, giving enough incentives for mapping to an optimal policy for the parents' problem. The elicitation process completed in 34 rounds with Δ as shown in Figure 4.9(b). The provided incentives induced the child to walk down and to the right from school until he reached home,



(a) child (b) motivation (c) induced policy

Figure 4.9: Domain without an expert. $D_{max} = 3.5$, $R_{max} = 30$, $\epsilon = 0.01$, $\lambda = 0.625$.

ϵ	rounds	$\widehat{GV}_{max}$	GV_{opt}	total time	time/round	last round time
0.01	34	2.1177	2.1177	64m 28s	1m 54s	9m 56s
0.1	11	2.1177	2.1177	17m 47s	1m 37s	9m 58s

Table 4.2: Domain without an expert. $D_{max} = 3.5$, $R_{max} = 30$, $\lambda = 0.625$.

never visiting the park in the process. The motivated policy is shown in 4.9(b), and achieves the optimal $GV(start) = GV_{opt}$.

Notice that the induced policy in Figure 4.9(b) still has the child going to the park when he is within one square of the park. Were the parents concerned about the child going to the park from states other than school, we can modify the objective of maximizing $GV(start)$ to maximizing the value in a number of states, with weights on each state based on the parents' expectations of where the child may start from at any given time.

We repeated the experiment with $\epsilon = 0.1$ to see what effect this change would have on speeding up the elicitation process. Our algorithm completed in 11 rounds with a mapping to the optimal policy for the parents' problem. This three fold decrease is impressive and again shows that setting ϵ values as large as possible is valuable for reducing the number of rounds of elicitation in progress. It also suggests that our heuristics and constraints are doing well for keeping the number of rounds of elicitation low even for the much smaller $\epsilon = 0.01$.

We present a comparison of the running time of the elicitation process based the two ϵ values in Table 4.2. In both cases, our algorithms did not immediately return with the best solution once it was found, but had to then prove that the mapping found was the best achievable across all remaining possible agent rewards. This last round is especially computationally expensive, taking approximately 10 minutes in both cases. It makes sense then to add an early stopping condition, such that if $\widehat{GV}_{max} = GV_{opt}$ we can stop elicitation knowing that there are no better mappings. More generally, the interested party may wish to add early stopping

conditions to end the elicitation process as soon as a found mapping induces a policy that matches a value threshold.

4.2.3 Summary

In our experiments, the implementations of Algorithm 1 and Algorithm 2 with the slack maximizing heuristics were able to solve the policy teaching problem after a small number of elicitation rounds. In domains with an expert, we saw that our chosen parameters performed well, and that the parameters themselves can have an effect on the length of the elicitation process. We also saw that our slack maximizing heuristics kept the number of elicitation rounds low, even for ϵ values that may otherwise lead to a much longer elicitation process. For domains without an expert, the elicitation process completed in a fairly low number of rounds, despite not having a fixed target value or policy to aim for. We saw that setting a larger ϵ is helpful for reducing the number of rounds of elicitation; this suggests that the interested party may wish to speed up the elicitation process by setting a larger ϵ and providing slightly more incentives to guarantee a mapping.

Chapter 5

Discussion

5.1 Applications

As we have stated in our introduction, there are many scenarios in which an interested party wishes to provide incentives to alter an agent's behavior. In most cases, the interested party does not know the agent's rewards, and is limited in the amount of incentives it can provide. We have introduced algorithms for policy teaching that address these issues. Our algorithms are based on a general MDP framework, which allows us to represent policy teaching problems in complex sequential decision tasks.

5.1.1 Real-World Applications

One particular area that policy teaching may be useful for is education. We can think of policy teaching as attempting to understand the underlying reasons of why a student performs a certain way, based on which motivation, incentives, and help can be provided to guide the student towards performing well. This approach allows for personalizing educational experience to individual agents by providing motivations that are specific to the preferences of the agent in question. In technologies, we can imagine using policy teaching in training simulators (i.e. a flight simulator) and educational software, in which the program learns about the

rewards that govern a particular student's behavior, based on which prompts and feedbacks can be provided to assist the student. In the classroom, we can imagine using policy teaching to understand how to motivate a student to solve problems correctly. Consider the following example:

Example 5. Student solving a problem
A student is presented with a problem. The student starts solving the problem without rereading the problem statement, writes down the first solution he comes up with, declares that he has finished, and goes to play in the playground. The student is not very careful, and only gets the correct answer half of the time. The teacher is interested in helping the student solve problems correctly, but has been unsuccessful in his attempts. The teacher is willing to offer healthy snacks as motivation, but has a limited budget.

We can solve this problem using the policy teaching framework. We use a MDP to capture the states, actions, and outcome probabilities of the student's actions. Figure 5.1 gives a state and action diagram for the student. The student is presented with a problem; he can reread the problem statement, which gives him a higher probability of answering the question correctly when he solves the problem. Based on the student's answer, he may either declare that he is done or check his answer. If the student chooses to check his work when he has the right answer, he has a high probability of declaring that he is done and a low probability for switching to the wrong answer. If the student has the wrong answer, checking gives him a high probability of switching the wrong answer to the right one, and a low probability for declaring that he is done. The child then moves to the "solved correctly" or "solved incorrectly" state depending on his answer, but goes to play in the playground regardless of the outcome.

Based on the problem definition, we can apply methods of policy teaching to learn the student's rewards and figure out what incentives to provide in what states to get the child to reach the "solved correctly" state more often. It is possible

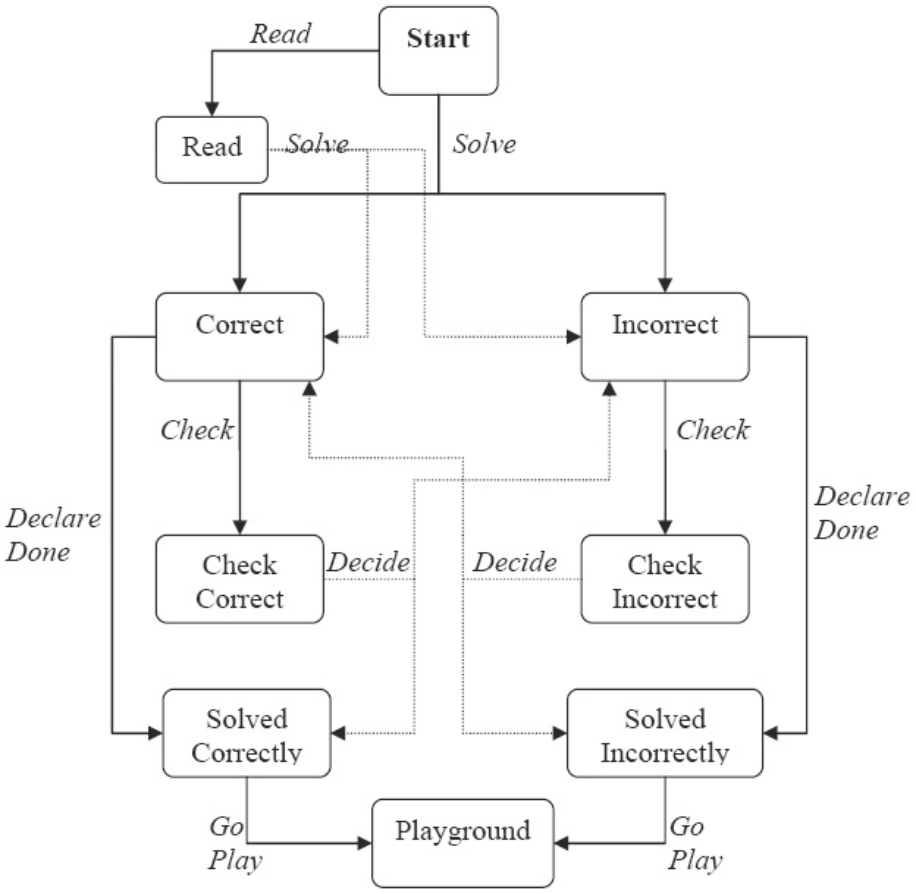


Figure 5.1: Example state space of a student solving a problem.

that the child sees rereading and checking as costs, and cares more about finishing quickly to get to the playground than solving problems correctly. We can provide the student with incentives and observed the induced policy, based on which to narrow our beliefs about the student's rewards until we find an admissible mapping to a policy with high value for the teacher's problem (wanting the child to solve problems correctly).

Another area of application for policy teaching is in economic settings, where cases of an interested party wishing to provide incentives for an agent to behave a certain way are commonplace. An employer may be interested in learning about

the preferences of an employee to figure out how to provide incentives for the employee to perform well on the job. A search engine may be interested in learning about the preferences of potential users, as to provide incentives for people to use its service. A store may be interested in learning about the preferences of its users, as to figure out how to provide incentives for users to buy products.

In each of these cases, the agent rewards may depend on many factors, some of which are monetary but many of which are not. Furthermore, an agent may take a number of actions before deciding to sign a contract or purchase a product, and influencing earlier decisions could have important effects on the final outcome. Policy teaching and its MDP framework can capture the dynamics of the process, using observations of interrelated actions in complex domains to reason about the agent's preferences.

In addition to solving the policy teaching problem itself, the methods we have developed may be useful for eliciting preferences in a complex domain and using the preferences to accomplish a different task. For example, in user interface design, understanding the preferences of a user of a particular interface may provide information about how to generate a better interface for that user. In policy teaching, we are concerned with making an agent perform well taking the conditions of the problem as given. In other cases, we may be interested in generating conditions under which the agents preferences perform well. The elicitation methods we have developed could be useful for learning agent preferences in these situations as well.

5.1.2 Multi-agent Systems

Techniques from policy teaching may have applications to the study of multi-agent systems, problems in which a social planner is interested in implementing an outcome that depends on the inputs of self-interested agents whose preferences are unknown to the social planner. This problem is well studied in the areas of mecha-

nism design and implementation theory, where the goal is to design a mechanism that provide the rules of interactions under which the agents' reports lead to the desirable outcome.

In many situations, coming up with a mechanism with the desired properties is a difficult task. This is especially true if the mechanism must work well regardless of the true preferences of the agents. Furthermore, mechanisms may be difficult to implement in practice. In domains with complex preferences, an agent may be unable to express its preferences accurately to meet the mechanism's demands.

Policy teaching provides an alternative method for implementing desirable outcomes. Instead of designing the rules of interactions, we may be able to provide minimal incentives to the agents to induce behaviors that lead to the implementation of desirable social outcomes. Instead of designing mechanisms that work well regardless of the agents' preferences, we may be able to learn the agents' preferences by providing incentives and observing the induced behaviors of agents. We can then provide incentives for these agents to behave as desired using a policy teaching framework, or use the learned preferences to aid the design of a mechanism for the agents in question.[1]

The single agent policy teaching framework we have introduced has natural extensions to multi-agent domains. Consider the following multi-agent policy teaching problem.

Definition 22. Multi-agent policy teaching in domains with experts
An interested party is concerned with the policies of a group of n agents. Each agent i faces a MDP $M^i = \{S^i, A^i, R^i, P^i, \gamma^i\}$, and performs its optimal policy π^i. The interested party has knowledge of the agents' MDPs, and can provide an admissible incentive function Δ such that agent i receives an incentive of $\Delta^i(s)$ in state s to modify the agent's reward function to $R^i + \Delta^i$. Can the interested party

[1]The idea of using minimal incentives to implement outcomes has been previously considered by Monderer and Tennenholtz [14]. Their work provided general results for normal form games, but do not deal with sequential decision tasks nor the issue of learning rewards.

find a Δ to induce every agent i to perform policy π_E^i?

This problem requires the agent policies to be considered in unison, but is otherwise no different than in the single agent case. We can write Δ constraints over all the agents, and map each agent i to an expert reward R_E^i that satisfies the IRL constraints on the expert's policy:

$$\min_{R_E} \sum_i \sum_s \Delta^i(s) \tag{5.1}$$

subject to:

$$\Delta^i(s) = R_E^i(s) - R^i(s) \qquad \forall s, i \tag{5.2}$$

$$(\mathbf{P^i_{\pi^i_E}} - \mathbf{P^i_a})(\mathbf{I} - \gamma^{\mathbf{i}}\mathbf{P^i_{\pi^i_E}})^{-\mathbf{1}}\mathbf{R^i_E} \succeq \epsilon \qquad \forall a \in A \backslash a1, a1 \in \pi(s), i \tag{5.3}$$

$$\sum_i \sum_s \Delta^i(s) \leq D_{max} \tag{5.4}$$

$$\Delta^i(s) \geq 0 \qquad \forall s, i \tag{5.5}$$

If the agents' reward functions are unknown to the interested party, we can still solve this problem by using a similar elicitation method as we use in the single agent case. At each iteration, we can add IRL constraints on R^i to Program 5.1, and use the linear program to find possible agents' rewards that have mappings to the set of expert policies.[2] We can then provide Δ to each of the agents, and observe the induced policy. If every agent i performs the expert policy π_E^i, we can complete the elicitation process with the mapping Δ. Otherwise, for the agents that did not behave as expected, we can add IRL constraints on their induced policies. This narrows the space of possible rewards for those agents. Since we are guaranteed to make progress at every iteration, we will eventually be able to find a mapping that induces each agent i to perform π_E^i, if such a mapping exists given the limits on Δ.

[2]We may also switch the objective (i.e. to maximizing the sum of the agents' slacks), if we wish.

The multi-agent policy teaching problem in Definition 22 is simple, but can apply to many settings in which an interested party is concerned with the joint policy of a group of agents. By framing the problem as one multi-agent policy teaching problem instead of many single-agent policy teaching problems, the interested party is able to find a way to distribute the incentives and find mappings that may otherwise have not been admissible were the interested party to set aside limits on the incentives provided to each agent. Further extensions of policy teaching to the multi-agent domain provide an interesting area for future research.

5.2 Critique

In this section we acknowledge the major critiques of our work, and evaluate the significance of these objections. We divide these critiques into five sections, and propose extensions to the policy teaching framework to address the issues raised.

5.2.1 Parameters and Constraints

- **The limits on Δ do not accurately capture the amount of incentives the interested party is willing to provide. Some states may be visited more than once, and other states may not be visited at all.**
  One can capture the history of states visited by constructing a MDP that models history into the state space. This allows the interested party to specify incentives that can only be received once, such that an agent receiving an incentive in a state will be transitioned to history states that take the information into account to prevent the agent from receiving the incentive again. However, this requires an exponential blowup in the state space, and is only feasible if the state space is small. Alternatively, we can introduce additional (or different) constraints to reflect that a state may be visited forever in reasoning about incentives. For example, we can constrain the total sum of discounted incentives that an agent can receive by specifying $\Delta(s)/(1-\gamma) < D_{max}, \forall s$.

We can also construct a belief over what states the agent will visit and how often, and tailor constraints on provided incentives based on learned beliefs.

In many cases, the admissible constraints on Δ are sufficient. Many problems can be modeled by a linear progression of states, where the transitions prevent states from being revisited. Other problems have very few states that can be revisited, and such states can be treated separately. In all problems modeled, it is unlikely that an interested party desires the agent to perform a policy in which an agent would be getting incentives over and over again. As long as the policy teaching problem is solved, the limits on Δ do accurately reflect the incentives that the interested party provides. The exception to this is goal states that the interested party wishes the agent to reach (i.e. "home" in the child walking home example), but these cases can either be modeled using absorbing states, or by taking into consideration that incentives provided in this state is for the discounted infinite future.

- **The minimal slack ϵ needs to be set high for elicitation to finish quickly. The amount of extra incentives that may be required for there be a mapping with ϵ slack is dependent on ϵ and the number of states n. If n is large, a large ϵ may prevent available mappings. This forces a smaller ϵ to be set, slowing the elicitation process.**

  This critique is a serious one, as it suggests that the length of elicitation can increase quickly for larger state spaces not only because of the increase in the space of possible rewards but due to the need to set a smaller ϵ. We can deal with this issue in a number of ways. First, the amount of extra incentives necessary for there to be a mapping is calculated in the worst case. There is likely to be other mappings that do not require much extra incentive to be provided for there to be a slack of ϵ. This suggests that we can greedily set ϵ to be large, and scale back if necessary. Second, when the state space is large, we may wish to assume some structure on the reward function, which can sig-

nificantly decrease the space of possible rewards that needs to be considered. Finally, as we have shown in our experiments, our slack maximizing heuristics allows our algorithm to scale well even for smaller ϵ. Improvements in elicitation heuristics and ways to ensure progress other than the slack based on ϵ can help to address this issue.

5.2.2 Assumptions

- **Policy teaching assumes the interested party knows the agent's MDP definition.**

  This assumption is not too restrictive. By watching the agent behave in the domain, the interested party can learn the transition probability of an agent moving from state to state based on his actions. The discount factor may also be learned, possibly by observing the change in the agent's responses to provided incentives in different time periods.

- **Policy teaching assumes that the interested party's problem is based on the same MDP$_{-R}$ definition as the agent.**

  We have framed the problem in this manner to give a sense of what the interested party is concerned with, but can represent the interested party's preferences generally as any function of the agent's policy, independent of the MDP$_{-R}$ definition.

- **Policy teaching assumes the interested party knows his own reward G.**

  In cases where an interested party cannot write down his preferences, he may still be able to specify policies that he is interested in the agent performing. In such cases, we can have the agent attempt to map to these policies using methods from policy teaching with an expert. In other cases, an interested party may be able to specify his rewards roughly (either directly or through elicitation), which can then be refined over time.

- **Policy teaching assumes that the expert has the same MDP$_{-R}$ definition as the agent.**

  We think of the expert as having a similar background as the agent, which makes having the same (or similar) MDP$_{-R}$ definition likely. For example, in the schooling setting, we think of the expert as a motivated student, not a teacher whose problem definition may be very different. Furthermore, the expert need not face the same MDP$_{R}$ as the agent; if we are only concerned with whether we can get the agent to perform like the expert, there need only exist an admissible mapping to the expert's policy.

- **Policy teaching assumes that the interested party can observe the agent's behavior in every state. In some situations, actions may be unobservable in certain states of interest.**

  An example of a scenario with unobservable action is when a service provider is interested in increasing its customer base by learning the preferences of people using competing services. If the service provider does not have access to the behaviors of users of competing services (neither policies nor trajectories), we cannot learn the rewards of these agents using policy teaching. However, we may be able to infer these agents' rewards in some other way in order to apply policy teaching with known rewards. We can also use policy teaching to learn the rewards of agents that are in the system, and use this information to infer about how to motivate other users to switch providers. While policy teaching does not deal with unobservable actions, its methods may still be useful for these situations.

5.2.3 Expressiveness

- **Rewards are placed on states, but in many cases it is more natural to express rewards on state-action pairs.**

  The current framework is easily extendable to reward functions over state-

action pairs. Furthermore, our elicitation algorithms do not rely on the form of the reward function.

- **Policy teaching is limited to finite state spaces.**
  We can represent infinite state spaces by assuming (or approximating) reward functions as a linear combination of known features, where the weights on these features represents the agent's unknown reward function. Ng and Russell [16] provided extensions to their IRL algorithms for infinite state spaces, which we can apply to learn agent rewards given a policy. We can find mappings to expert policies by providing incentives with Δ constraints based on the guessed weights and features. New evidence can then be incorporated in the same manner.

- **Policy teaching requires the interested party to have access to the agent's policy. More realistically, the interested party may only see a set of actual trajectories of the agent's actions in the state space.**
  Ng and Russell [16] provides IRL algorithms for learning reward functions based on trajectories. The extension is similar to IRL for infinite state spaces, and can be incorporated into the policy teaching framework in much the same way.

- **Policy teaching does not incorporate priors over possible agent rewards.**
  Ramachandran and Amir [21] extended the IRL framework to Bayesian IRL (BIRL), and presented algorithms for combining prior knowledge and observations to derive a probability distribution over the space of possible rewards. To allow for priors in policy teaching, we can use BIRL instead of IRL to infer rewards. The distribution over possible rewards is useful for choosing what to guess for the agent reward, and can lead to a shorter elicitation process. In cases where the interest is in learning the true reward function, Bayesian updating over possible rewards can also provide more detailed information about the agent's preferences.

The prior-based preference elicitation literature on sequential decision tasks may provide further ideas for how to use probabilistic distributions over reward functions for policy teaching. Chajewska, Koller, and Parr [10] developed methods for finding queries that lead to the most information gain based on a distribution over possible utility functions. Chajewska, Koller, and Ormoneit [9] studied the problem of learning utility functions based on prior probability distribution and observed behavior. Understanding how these works fit within the policy teaching framework may be useful for policy teaching with priors.

5.2.4 The Elicitation Process

- **Algorithms 1 and 2 complete without guessing the agent's reward correctly. These methods are useful for policy teaching, but not for learning the agent's rewards.**

  While our elicitation process may terminate before the true agent reward is found, the process narrows the space of possible agent rewards. All rewards that remain in the set of possible rewards "explains" the agent's behaviors, as they all satisfy the IRL constraints on the induced agent policies. Further improvements are possible by incorporating a probability distribution over rewards as we have described, but policy teaching as demonstrated does plenty towards learning the agent's true reward. An interesting open question is whether we can use perturbations to the agent's rewards to narrow the space of possible rewards down to the agent's true reward. One possible approach is to figure out policies other than the target policy to map the agent's reward, so as to be able to further narrow the space of rewards.

- **The constraints on the agent's reward are based on the values of the incentives provided. Values found this way do not reflect the agent's true reward. Also, the interested party may not be able to accurately map the**

relation between different types of incentives.

We answer each part of the critique in turn. First, reward functions are mappings of preference relations; we are not interested in cardinal utility theory here, and do not assign significance to the magnitude of the differences between rewards. While we have been saying the agent's "true" reward throughout this work, we mean by this the reward function that expresses the agent's preferences with respect to the values used to represent the incentives. Any size comparisons we make in our discussion is with respect to values assigned to the incentives. In domains where the incentives provided are of the same form (i.e. all monetary), studies in economics provides tools for modeling the relative values of different incentives.

In the case where different types of incentives are provided, we need to come up with the relation between these types of incentives based on which to constrain the agent's reward. If the relation we come up with is not the way that the agent perceives the relationship, the constraints we add may not be very meaningful in expressing the agent's preference space. One possible solution is to treat the relationship as unknown and attempt to learn it. This can be done outside of the policy teaching framework, or as part of the framework where we consider one type of an incentive as an unknown function of another. We can add constraints based on induced policies in the same way, but with the incentive Δ added expressed as a vector variable based on the unknown function instead of a vector of values. We may also maintain a probabilistic distribution over possible relation functions, and use observations and other evidence to update the distribution. Such solutions may introduce additional computational complexity, and figuring out how to do so generally but tractably is an interesting topic for future research.

- **The policies the agent performs are assumed to be optimal for the agent's reward function. This may not be the case.**

While policy teaching attempts to find a reward function that correspond to the agent's policies, the agent's reward function may be changing over time. Another possibly is that the agent's rewards are constant, but the way he reasons about them changes over time. Without a model how how the agent's reward evolves over time nor an accurate model of the agent's reinforcement learning process, placing IRL constraints on the agent's reward may be too restrictive. One possibly is to express the IRL constraints as soft constraints, for which we wish to satisfy as closely as possible but allow for some constraints to be violated. Soft constraints on utility functions have proved to be useful in preference elicitation for user interface generation [11, 12], and could be useful for policy teaching as well.

5.2.5 Long Term Effects

- **If the agent's problem definition remains constant, the interested party must provide incentives for the agent to behave as desired every time the agent performs the sequential decision tasks.**
  In the teaching, parenting, and advertising examples that motivate our work, we imagine that an interested party believes that the agent "absorbs" the provided incentives over time, such that the agent will perform as the interested party desires without the interested party having to provide much (or any) additional incentives in the future. This belief appears to be accurate in many real-world situations, where an agent may form habits and stick to his induced policy due to fixed costs for switching to another policy (possibly the same fixed costs that kept the agent away from the policy desired by the interested party in the first place). The interested party may also try to gradually retract the amount of incentives provided, until he finds the minimal incentive based on which the agent will still behave as desired. In other cases, the interested party may be okay with providing incentives every time

the agent performs the tasks. For example, consider a consumer who only buy products when they are on sale. A firm is happy to provide discounts for the consumer every time as long as the firm can sell the good and make a profit.

- **Providing large incentives for an agent to perform as desired may have undesirable long term effects.**

  For large incentives, the agent may not be able to motivate himself enough to internalize the incentives provided. Furthermore, large incentives may lead to the agent performing as the interested party desires only because of the incentives provided. For example, if a student wishes to solve problems correctly only because of the candy provided, the student may have no desire to solve the problem correctly were the candy taken away. Just as the interested party wishes for the incentives to be absorbed, the provided incentive may dictate the agent's behavior such that the agent's inherent rewards wither over time.

  While this is not a direct critique of policy teaching but of the effect of incentives on people over time, it is an important issue that is worth studying further. We may be able to model the long term effects of incentives, and take such effects into consideration when providing incentives to agents. However, in many situations, the amount of incentives that can be provided is limited, and only enough to motivate an agent towards decisions that he has inherent values for. For example, we do not expect a firm to provide discounts so large as to cause consumers to not make purchases unless steep discounts were applied in the future. Nevertheless, understanding the dynamics of how an agent's preferences and problem definition changes over time can be valuable for policy teaching.

5.3 Open Questions and Future Research

In addition to potential applications and extensions, our work has created many open questions for future research on the current policy teaching framework. They include:

- Allowing for the provision of punishments. What effect does this have on what policies can be induced? Does this allow for more effective elicitation strategies? This may also be useful for extensions to multi-agent systems, where having punishments in addition to rewards allows for transfers among agents.

- Improvements to objectives for choosing an agent reward. The effectiveness of the objective determines whether the elicitation process can solve the policy teaching problem in a small number of rounds. Can we exploit the structure of certain domains, or incorporate prior information?

- Conduct more experiments on simulated and real-world examples. Evaluate the effect of parameters on the length of the elicitation process. How well do the current parameter setting methods perform? Attempt to better understand why certain values work well (or poorly), and use found understanding to create better methods for parameter setting.

- Attempt to come up with more tractable formulations for policy teaching without an expert. Can we find approximate solutions quickly?

- Generalize policy teaching with an expert. Instead of mapping to a specific policy, how can we have the agent map to any policy within a set of policies?

Chapter 6

Conclusion

How can an interested party provide incentives to affect the behavior of an agent?

This question comes up in economics. This question comes up in education. This question comes up in parenting. This question comes up whenever there is a conflict of interest and one party wishes to affect the behavior of another party. This question comes up in real-world examples, and in technical problems.

This is a hard problem to solve. It requires understanding how provided incentives and inherent preferences are related to behavior. It requires understanding how provided incentives and inherent agent preferences are related to each other. It requires being able to model the agent's behavior in complex domains. It requires taking the limits on the incentives that the interested party can provide into account. It requires knowledge of the preferences of the parties involved. When preferences are unknown, solving the problem requires learning the preferences.

This thesis developed a general framework for studying this problem that takes these requirements into account.

6.1 Brief Review

In Chapter 1, we introduced real-world scenarios in which an interested party wishes to affect the behavior of an agent by providing incentives. We outlined the

major obstacles that the interested party faces in trying to get an agent to behave as desired, and provided a non-formal definition of the policy teaching problem. We described our contributions for studying the problem in a Markov Decision Process (MDP) framework, and discussed our work in the context of related works in contract theory, apprenticeship learning, and preference elicitation.

In Chapter 2, we motivated and developed the MDP formalism for modeling sequential decision tasks. We used the framework to provide a formal definition of the policy teaching problem with known rewards. We studied the problem in domains without an expert, and formulated a mixed integer program as a solution to a difficult technical problem. We then considered the policy teaching problem in domains with an expert, where our solution is a simple linear program.

In Chapter 3, we introduced techniques of inverse reinforcement learning (IRL) for finding the space of reward functions to corresponds to a policy. We used IRL to generalize our solution for domains with an expert, both to allow for more potential mappings and domains where the agent's reward is known but the expert's reward is unknown. We then introduced the policy teaching problem with unknown agent rewards, and realized that while IRL provided an useful starting point, we required additional evidence about the agent's reward to solve the policy teaching problem. We discussed why direct utility elicitation methods may not apply well to policy teaching, and proposed a method for generating additional evidence about the agent's rewards based on observations of the agent's responses to provided incentives. We turned this method into an elicitation algorithm for policy teaching in domains with an expert, and proved bounds on the number of elicitation rounds the algorithm requires before returning a solution. We extended the ideas to domains without an expert, constructed a similar algorithm for such domains, and proved similar bounds on the number of elicitation rounds. Having presented general elicitation algorithms for policy teaching with unknown reward, we discussed possible objective functions for the elicitation process, and

introduced tractable heuristics for reducing the number of elicitation rounds in practice.

In Chapter 4, we discussed experiments on using our algorithms with the heuristics we had introduced. We described methods for setting the various parameters of the algorithms which may allow for a shorter elicitation process. We presented results on a motivating example, and showed that we can complete the elicitation process in a small number of rounds for both domains with and without an expert. For the domain with an expert, we studied the effects of setting parameters, and determined that while our set parameters performed well, improvements are possible. For the domain without an expert, we saw that while we were able to find a solution to the policy teaching problem for our example, the computational complexities introduced by the mixed integer programs may become a bottleneck for larger domains.

In Chapter 5, we discussed the potential applications of our work, both for real-world settings and for multi-agent system design. We presented the major critiques of our work, which we divided into sections on parameters and constraints, assumptions, expressiveness, elicitation process, and long term effects. We proposed extensions to address these critiques, and concluded the chapter with open questions on the policy teaching framework for future research.

6.2 Conclusion

Our work provides a general framework for studying the policy teaching problem. The policy teaching algorithms we have introduced provide a new way for eliciting preferences that draws from the connection among behaviors, provided incentives, and inherent preferences. Our policy teaching framework provides a solid foundation for possible extensions and applications, and offers exciting open questions for future research.

References

[1] Peter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

[2] Joseph Beck, Mia Stern, and Erik Haugsjaa. Applications of AI in education. *Crossroads*, 3(1):11–15, 1996.

[3] Jennifer Boger, Pascal Poupart, Jesse Hoey, Craig Boutilier, Geoff Fernie, and Alex Mihailadis. A Planning System Based on Markov Decision Processes to Guide People with Dementia Through Activities of Daily Living. *IEEE Transactions on Information Technology in Biomedicine*, 10(2):323–333, 2006.

[4] Patrick Bolton and Mathias Dewatripont. *Contract Theory*. MIT Press, 2005.

[5] Craig Boutilier. POMDP Formulation of Preference Elicitation Problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pp. 239–246, 2002.

[6] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based Optimization with the Minimax Decision Criterion. *Ninth International Conference on Principles and Practice of Constraint Programming*, pp. 168–182, 2003.

[7] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Regret-based Utility Elicitation in Constraint-based Decision Problems. In *Proceedings*

of the Nineteenth International Joint Conference on Artificial Intelligence, pp. 929–934, 2005.

[8] Craig Boutilier, Tuomas Sandholm, and Rob Shields. Eliciting Bid Taker Non-price Preferences in (Combinatorial) Auctions. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pp. 204–211, 2004.

[9] Urszula Chajewska, Daphne Koller, and Dirk Ormoneit. Learning an Agent's Utility Function by Observing Behavior. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[10] Urszula Chajewska, Daphne Koller, and Ronald Parr. Making Rational Decisions using Adaptive Utility Elicitation. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pp. 363–369, 2000.

[11] Krzysztof Z. Gajos, Jing Jing Long, and Daniel S. Weld. Automatically Generating Custom User Interfaces for Users with Physical Disabilities. In *Proceedings of the Eight International ACM SIGACCESS Conference on Computers and Accessibility*, pp. 243–244, 2006.

[12] Krzysztof Z. Gajos and Daniel S. Weld. Preference Elicitation for Interface Optimization. In *Proceedings of the eighteenth annual ACM symposium on User interface software and technology*, pp. 173–182, 2005.

[13] Jean-Jacques Laffront and David Martimort. *The Theory of Incentives: The Principal-Agent Model*. Princeton University Press, 2001.

[14] Dov Monderer and Moshe Tennenholtz. k-Implementation. In *Proceedings of the fourth ACM conference on Electronic Commerce*, pp. 19–28, 2003.

[15] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 1999.

[16] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[17] Relu Patrascu, Craig Boutilier, Rajarshi Das, Jeffrey O. Kephart, Gerald Tesauro, and William E. Walsh. New Approaches to Optimization and Utility Elicitation in Autonomic Computing. In *Proceedings of the Twentieth national Conference on Artificial Intelligence*, pp. 140–145, 2005.

[18] Bob Price and Craig Boutilier. A Bayesian Approach to Imitation in Reinforcement Learning. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 712–717, 2003.

[19] Bob Price and Craig Boutilier. Accelerating Reinforcement Learning through Implicit Imitation. *Journal of Artificial Intelligence Research*, 19:569–629, 2003.

[20] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, New York, NY, 1994.

[21] Deepak Ramachandran and Eyal Amir. Bayesian Inverse Reinforcement Learning. *Twentieth International Joint Conference on Artificial Intelligence*, 2007.

[22] Tuomas Sandholm and Craig Boutilier. Preference Elicitation in Combinatorial Auctions. *Combinatorial Auctions, Chapter 10.* MIT Press, 2006.

[23] Aaron P. Shon, David B. Grimes, Chris L. Baker, and Rajesh P.N. Rao. A Probabilistic Framework for Model-Based Imitation Learning. In *Proceedings of CogSci*, 2004.

[24] Hal R. Varian. Revealed Preference. *Samuelsonian Economics and the Twenty-First Century, Chapter 5.* Oxford University Press, 2007.

[25] Tianhan Wang and Craig Boutilier. Incremental Utility Elicitation with the Minimax Regret Decision Criterion. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 309–316, 2003.