Title
The Principal-Agent Alignment Problem in Artificial Intelligence

Author
Hadfield-Menell, Dylan Jasper

Publication Date
2021

Peer reviewed|Thesis/dissertation

The Principal–Agent Alignment Problem in Artificial Intelligence

by

Dylan Jasper Hadfield-Menell

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Stuart J. Russell, Co-chair
Professor Anca D. Dragan, Co-chair
Professor Pieter Abbeel
Professor Ken Goldberg

Summer 2021

The Principal–Agent Alignment Problem in Artificial Intelligence

Abstract

The Principal–Agent Alignment Problem in Artificial Intelligence

by

Dylan Jasper Hadfield-Menell

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Stuart J. Russell, Co-chair

Professor Anca D. Dragan, Co-chair

The field of artificial intelligence has seen serious progress in recent years, and has also caused serious concerns that range from the immediate harms caused by systems that replicate harmful biases to the more distant worry that effective goal-directed systems may, at a certain level of performance, be able to subvert meaningful control efforts. In this dissertation, I argue the following thesis:

1. The use of incomplete or incorrect incentives to specify the target behavior for an autonomous system creates a *value alignment problem* between the principal(s), on whose behalf a system acts, and the system itself;
2. This value alignment problem can be approached in theory and practice through the development of systems that are responsive to uncertainty about the principal's true, unobserved, intended goal; and
3. Value alignment problems can be modelled as a class of cooperative *assistance games*, which are computationally similar to the class of partially-observed Markov decision processes. This model captures the principal's capacity to behave strategically in coordination with the autonomous system. It leads to distinct solutions to alignment problems, compared with more traditional approaches to preference learning like inverse reinforcement learning, and demonstrates the need for strategically robust alignment solutions.

Chapter 2 goes over background knowledge needed for the work. Chapter 3 argues the first part of the thesis. First, in Section 3.1 we consider an order-following problem between a robot and a human. We show that improving on the human player's performance requires

that the robot deviate from the human's orders. However, if the robot has an incomplete preference model (i.e., it fails to model properties of the world that the person does care about), then there is persistent misalignment in the sense that the robot takes suboptimal actions with positive probability indefinitely. Then, in Section 3.2, we consider the problem of optimizing an incomplete proxy metric and show that his phenomenon is a consequence of incompleteness and shared resources. That is, we provide general conditions under which optimizing any fixed incomplete representation of preferences will lead to arbitrarily large losses of utility for the human player. We identify dynamic incentive protocols and impact minimization as theoretical solutions to this problem.

Next, Chapter 4 deals with the second part of the thesis. We first show, in Section 4.1, that uncertainty about utility evaluations creates incentives to get supervision from the human player. Then, in Section 4.2 and Section 4.3, we demonstrate how to use uncertainty about utility evaluations to implement reward learning approaches that penalize negative side-effects and support dynamic incentive protocols. Specifically, we show how to apply Bayesian inference to learn a distribution over potential true utility functions, given the observation of a proxy in a specific development context.

Chapter 5 deals with the third part of the thesis. We introduce *cooperative inverse reinforcement learning* (CIRL), which formalizes the base case of *assistance games*. CIRLmodels dyadic value alignment between a human principal **H** and a robot assistant **R**. This game-theoretic framework models **H**'s incentive to be pedagogic. We show that pedagogical solutions to value alignment can be substantially more efficient than methods based on, e.g., imitation learning. Additionally, we provide theoretical results that support a family of efficient algorithms for CIRL that adapt standard approaches for solving POMDPs to compute pedagogical equilibria.

Finally, Chapter 6 considers the final component of the thesis, the need for robust solutions that can handle strategy variation on the part of **H**. We introduce a setting where **R** assists **H** in solving a multi-armed bandit. As in Section 3.1, **H**'s actions tell **R** which of the k different arms to pull. However, this introduces the complication that **H** does not know which arm is optimal a priori. We show that this setting admits efficient strategies where **H** treats their actions as purely communicative. These communication solutions can achieve optimal learning performance, but perform arbitrarily poorly if the encoding strategy used by **H** is misaligned with **R**'s decoding strategy.

We conclude with a discussion of related work in Chapter 7 and proposals for future work in Chapter 8.

Contents

Acknowledgments

I am deeply grateful to my advisors for their guidance, support, and patience throughout this process. It has been an honor to be your student and I look forward to the opportunity to collaborate again in the future. Thank you for helping shape me into the researcher that I am today.

To Prof. Anca Dragan, thank you for the joy you brought to research meetings, for teaching me to be experimentally rigorous, and for sharing your insight into the nature of practical, algorithmic, human-robot interaction. To Prof. Stuart Russell, thank you for teaching me to do careful research, for always taking the time to discuss the bigger picture behind a research project, and for your guidance on how to write clearly and persuasively. To Prof. Pieter Abbeel, thank you for the opportunity to develop my skills as a mentor, for your support in that process, and for pushing me to go beyond my research comfort zone.

I would also like to thank Prof. Ken Goldberg for serving on my committee and for inviting me to collaborate with him and his students. I am also indebted to the students (undergraduate and graduate) that I have had the opportunity to work with. Thank you for making research fun and helping me to take breaks when I needed to. Thank you to the staff at the Center for Human-Compatible AI and the Berkeley AI Research Lab for their assistance managing the details of research.

Thank you to my family for their support and flexibility throughout graduate school. I know it was difficult at times, but I could not have done it without your support. Last, and most importantly, thank you to Veronica, for your commitment to my dreams and for the work to help me get there.

This dissertation collects and combines results developed in collaboration with, in addition to my advisors, Lawrence Chan, Jaime K. Fisac, Gillian K. Hadfield, Dhruv Malik, Smitha Milli, Malayandi Palaniappan, Ellis Ratner, Siddhartha Srinivasa, and Simon Zhuang. I am grateful to each and every one for their contributions. Portions of this text have been adapted from Chan et al. [31], Malik et al. [104], Hadfield-Menell and Hadfield [67], Milli et al. [109], Ratner, Hadfield-Menell, and Dragan [124], Zhuang and Hadfield-Menell [180], and Hadfield-Menell et al. [70, 73]. The terminology for pedagogic equilibrium was initially introduced in Fisac et al. [46]. This work was supported by a Berkeley Fellowship, National Science Foundation Graduate Research Fellowship Grant No. 1106400, the Center for Human-Compatible AI, FLI, OpenPhil, OpenAI, ONR, NRI, AFOSR, and DARPA.

Chapter 1

Introduction

1.1 The alignment problem in artificial intelligence

The field of artificial intelligence has made remarkable progress in the ability to program computers with goal-driven behaviors in response to visual, linguistic, and other perceptual input [179]. This includes more abstract achievements such as superhuman performance in zero-sum games [144, 110] and deployed commercial products such as data-driven recommendation systems that direct and monetize attention online [50, 174]. At the same time, the deployment of these systems has been accompanied by several high-profile failures and increasingly vocal concerns.

When we look at these examples, we can identify two types of failures. The first is an old story in the development of new technology: the system designers over-promised and the algorithm did not perform as intended. For example, consider the instances of fatal car accidents caused by Tesla's autopilot system [21]. In principle, better computer vision techniques, in the sense of more effective optimization of empirical and 'true' risk, can reduce the occurrence of these types of failures.

In the other cases, however, the failure stems from the system *effectively* optimizing for its stated objective. Consider the engagement optimization techniques used in recommendation systems [158]. These systems have come under serious scrutiny for their propensity to promote polarizing [16], addictive [76, 10], or extremist [79] content. It is clear that these harms are not a result of a machine learning failure — instead they are a good example of the system optimizing a narrowly defined objective effectively.[1] The propensity of optimized behavior to produce surprising, unintended results, has been observed in economic interactions [85] and autonomous systems [90].

[1]It is important to note that the problems observed in online platforms go beyond algorithmic challenges. Even with the ability to define the goal of 'good' engagement, there are regulatory challenges in getting companies to adopt those definitions and ethical questions to consider about who should provide such definitions. For this work, the core point is that we do not yet have clear techniques for how to provide these definitions.

In these situations, the *goal* specified by the system designer is faulty. Thus, improvements in the capabilities of AI systems, measured against their ability to optimize a given task, will not reduce the risk of these harms. In fact, we can expect that progress along the current trajectory of AI technology will *increase* the costs of misspecification for two reasons: First, more capable systems are more likely to be deployed in consequential settings where they can cause harm. Second, they will be more effective at exploiting gaps between a specified objective and the intended goal.

In this dissertation, I will argue that this type of failure is a direct consequence of the design pattern that has become dominant in the field of artificial intelligence. We will refer to this class of failures where the system malfunctions because it optimizes its objective too effectively as *alignment* failures. While misspecification is a challenge in any engineering application, I will argue that the nature of incentives and optimal behavior make this problem especially challenging and consequential in artificial intelligence. My thesis consists of three components:

1. The use of incomplete or incorrect incentives to specify the target behavior for an autonomous system creates a *value alignment problem* between the principal(s), on whose behalf a system acts, and the system itself;

2. This value alignment problem can be approached in theory and practice through the development of systems that are responsive to uncertainty about the principal's true, unobserved, intended goal; and

3. Value alignment problems can be modelled as a class of cooperative *assistance games*, which are computationally similar to the class of partially-observed Markov decision processes. This model captures the principal's capacity to behave strategically in coordination with the autonomous system. It leads to distinct solutions to alignment problems, compared with more traditional approaches to preference learning like inverse reinforcement learning, and demonstrates the need for strategically robust alignment solutions.

1.2 Incentives as a programming language for behavior

Before we discuss the problems caused by the use of incentives in the design of AI systems, it is useful to consider, first, why they have become the dominant paradigm in the field. One of the core challenges in the design of 'intelligent' autonomous systems is the need to balance three competing goals:

1. The program class used to represent the behaviors needs to be flexible enough that it can represent the desired behavior.

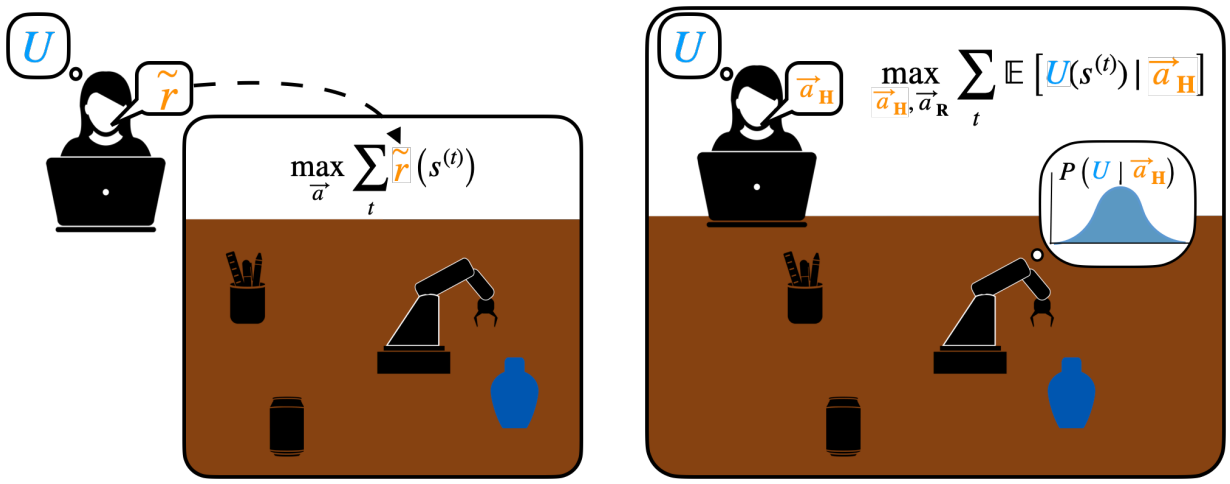


Figure 1.1: This dissertation formalizes the principal–agent alignment problem in artificial intelligence. **Left**: Most modern approaches to the design of autonomous systems rely on incentives to specify desired behavior. When these incentives are incomplete or incorrect, this creates a misalignment between the robot's objective, depicted as a proxy metric $\widetilde{r}$, and the principal's intended goal, depicted as $\mathcal{U}$. We argue in Chapter 3 that this misalignment can easily lead the system to perform poorly, perhaps catastrophically so. **Right**: A depiction of *cooperative inverse reinforcement learning* (CIRL). CIRL, which we formalize in Chapter 5, models value alignment as a cooperative *assistance game* where the true objective is unobserved by the robot $\mathbf{R}$ and is revealed through the principal's actions depicted as $\vec{a}_{\mathbf{H}}$. Crucially, CIRL models the principal's incentive to behave pedagogically (i.e., teach $\mathbf{R}$) and leads to distinct solutions to alignment problems, compared with more traditional approaches to preference learning such as inverse reinforcement learning.

2. The program class used needs to be simple enough that a real person can, directly or indirectly, specify an acceptable behavior out of the large set of alternatives.

3. The program representation needs to be such that it can be executed on high-dimensional perceptual input under time, energy, and other computational constraints.

While general-purpose programming languages are flexible enough that they could, in theory, represent a wide variety of interesting behaviors, it is simply too difficult for a human to directly encode a policy that, e.g., reliably navigates a mobile robot through a cluttered environment. Similar statements apply to language processing, high-dimensional motor control, and other subareas of artificial intelligence. One way to understand the issue is that the vast majority of programs exhibit highly uninteresting behavior when run on realistic perceptual input. Getting a system to do anything that resembles coherent behavior at all is a huge challenge.

In response to these challenges, the field of artificial intelligence developed a design pattern that lowers the burden on the designer. First, the field undertook a serious computational study of rational decision making and utility maximization. This allowed researchers to shift from the problem of specifying behaviors to specifying goals and allowing a general-purpose optimization algorithm to identify the correct actions to take. In effect, this prevents system designers from specifying behaviors that are useless: if the optimization algorithm runs correctly, then, at a minimum, any behavior executed will accomplish *some* objective.

The second major development was the adoption of statistical methods and probabilistic inference. It is not enough to program goal-achieving behaviors in an abstract model of the world. Mismatches between the abstract model and reality inevitably lead to undesired behavior. This is exemplified in robotics applications, where, e.g., uncertainty about the dynamics of the world and the impact of different controls on the world state make it hard to directly specify an accurate model. Probabilistic inference provided AI researchers with the ability to adapt models to observed data. Although much work remains to be done, these two developments give the designers of AI systems a plausible method to implement programs that execute coherent behaviors in the real world. The problem of 'programming' an AI system is then reduced to the problem of specifying the correct goal as, e.g., an empirical loss function for supervised learning or a reward function for policy optimization.

As a result, the field has invested heavily in the development of tools that function analogously to compilers. We have developed representation languages for incentives that, although not directly executable, can be converted into executable programs in a general-purpose manner. Just as a compiler takes an abstract representation of a program (e.g., in C code) and produces an executable program (e.g., machine code), a supervised learning algorithm takes an abstract representation of behavior (e.g., a dataset of images labelled cat or dog) and produces an executable representation of that behavior (e.g., a set of neural network weights for an image classification network).

At this point, we have built existence proofs of effective compilers. However, our research problems are analogous to benchmark code snippets for compiler research. As AI systems leave the lab, we will need to develop new compilers that can handle and support effective development environments. What properties make for a good behavior compiler? How do we build robust development practices for these software systems? How do we make sure they provide value for individuals and society?

It is clear that the specification of a 'good' compiler in this context includes much more than its ability to accomplish a specified goal. Let us return to the content recommendation example. The system's goal is to find 'good' content, in this case defined as content people will interact with. While this is reasonable in many cases, it has become clear that a lot of the most engaging content on the internet has undesirable properties. In this sense, we can interpret the system's misbehavior as a result of optimizing its objective 'too much.'

We will use the term *overoptimization* to refer to situations like this, where optimized behavior produces unwanted outcomes that score well according to the specified objective but not the true objective. This has long been observed as a property of incentive-driven behavior in humans. Kerr [85] discusses "the folly of rewarding A, while hoping for B" in a

variety of social settings. One salient example is that of organized cheating in universities, where students are rewarded for good grades while society hopes that they will gain true knowledge. Goodhart [59] and Campbell [26] observed similar issues with respect to public policy where market actors demonstrate the ability to game policy metrics and subvert intended policy goals. Strathern [157] articulated this phenomenon well in the context of teaching quality audits in British universities: "once a measure becomes a target, it ceases to be a good measure."[2]

In the study of artificial intelligence, overoptimization has received relatively little study, with one large exception: the classic machine learning problem of overfitting. Overfitting occurs when, e.g., a supervised learning system selects a classifier that reflects the noise in the data instead of the underlying concept the designers care about. This leads to high performance on the training data (i.e., effective optimization of the specified goal) but poor performance on new data. From an alignment perspective, we can understand this as a capable system that learns to exploit the gap between the goal we are able to specify (empirical risk) and the goal we actually care about (true risk on new data). Results that prove the consistency of learning methods can be understood as identifying a family of finite-size incentives (i.e., labelled datasets) that, in the limit of infinite size, converge to true classifier performance on new data.

Note that this exploitation of the gap between the specified and the true goal is not done maliciously — the system is not explicitly 'trying' to game the objective. Indeed, the true goal is unobserved and so these gaps are invisible to the system. It is simply the case that these gaps exist and, eventually, they will be leveraged in pursuit of the specified goal. As the capabilities of systems increase (defined as the ability of the system to optimize its incentives) the costs of misspecified incentives go up. At its most extreme such a system may represent an existential risk to human existence and flourishing [20]. If machines reach or exceed human abilities to accomplish open-ended goals, then the potential for conflict between a misspecified goal and, e.g., our continued existence and utilization of resources is a grave concern. However, one need not go to such extremes to appreciate the harms that overoptimizing a proxy metric can cause [165].

We propose to study this problem as a two-player decision problem with shared goals and asymmetric information about the goal. In this framework of *assistance games*, uncertainty about the goal plays a central role in determining the robot player's optimal action in any given situation. If the human actor is treated as a strategic player, this opens up a range of solutions. This assistance game formulation specifies incentives for the human player to optimally communicate their goals. As a result, we call solutions *pedagogic* equilibria. These solutions can be more effective, in the sense that the human–robot team generates more utility for the human player. However, communication-optimized strategies introduce the potential for coordination failures. Thus, solution concepts for assistance games should account for robustness to the human's strategy or effective ways to coordinate on a commu-

[2]This is often referred to as 'Goodhart's Law', although perhaps it should be renamed 'Strathern's Law' or 'Marilyn's Maxim.'

nication policy.

In the next sections, we provide an informal overview of the arguments behind the thesis.

1.3 Incompleteness and overoptimization

In Chapter 3, we make the case that overoptimization arises as a natural consequence of incomplete incentives. We begin with an analysis of an order-following robot **R** that learns the preferences of a principal, the Human **H**, on whose behalf it is supposed to act. We show that, in this model, **R** performs poorly if its preference model is incomplete in the sense that it does not represent features of the world that **H** cares about.

As an example, consider a coffee-making robot. Each morning, **H** looks at a menu of coffee drinks that can be made and selects their preferred choice. **R** observes this selection and prepares a beverage for **H**. We will assume that, like many humans who have not yet had their morning coffee, **H** is not perfect at making this choice. Thus, **R** can provide more utility for **H** by observing **H**'s choices over time and learning to correct for mistakes.

However, **R** is only able to increase **H**'s utility if **R**'s model of what **H** cares about is accurate enough. Consider the system behavior when **H** is trying to manage weight gain through a weekly calorie budget. They typically prefer high-calorie beverages but, depending on their expected intake of calories from other foods, they occasionally choose coffee options with fewer calories. Over time, the system 'learns' that these low-calorie choices are mistakes because it does not observe the calorie budget. It begins to make high-calorie options exclusively, subverting **H**'s weight-management goals.

Section 3.1 models this type of misalignment and shows that incomplete models can lead **R** to perform worse than **H** would on their own. Next, in Section 3.2, characterize situations where overoptimization is to be expected. We consider a model where utility can be decomposed into a monotonic function of several *attributes*. We show that overoptimization arises from the combination of incomplete incentives (i.e., a proxy objective that only references a subset of attributes), shared resources between attributes, and decreasing marginal utility per attribute.

We can illustrate this with the coffee-making robot. Instead of picking a drink from a menu each morning, **H** writes down a proxy metric that ranks different beverages and **R** uses this as an optimization target to purchase coffee beans, select brewing techniques, and mix ingredients. At some point, in order to devote more purchasing power to fancy brewing equipment, **R** discovers a black market of coffee bean producers that use environmentally destructive practices. **R**'s objective does not penalize illegal behavior or environmental damage, so this is optimal from the perspective of its specified objective. Despite this, **H** ends up with less utility because the marginal improvements in flavor are not worth the associated costs.

Section 3.2 provides technical conditions under which this will occur for *any* fixed and incomplete proxy metric. Section 3.3 considers relaxations of these conditions where **H** can penalize a generic measurement of impact, introducing weak dependence on the full attribute

set, or dynamically update **R**'s proxy metric. We show that both methods can, in theory, align **R**'s incentives with **H**'s enough to guarantee improvement from a starting condition.

1.4 Building systems that respond to uncertainty about goals

In Chapter 4, we make the case that a crucial component of the incentives for optimal assistance is **R**'s uncertainty about the utility evaluation for different states. Section 4.1 analyzes a model of oversight, where **R** has identified an action to potentially execute (e.g., buying coffee from the black market to cut costs). We analyze **R**'s incentives when faced with three options: 1) seeking oversight, where **R** shows **H** the action and gives **H** the ability to prevent execution by, e.g., turning **R** off; 2) directly acting, where **R** executes the action immediately, without communicating to **H** (i.e., bypassing oversight); or 3) turning off, where **R** decides to shut down on its own (i.e., without input from **H**).

We show that **R** will perceive the oversight action to be suboptimal if **R** 'knows' (i.e., believes it knows) the associated utility evaluations with certainty. Alternatively, if **R** is *uncertain* about utility evaluations and believes that **H** takes actions in response to expected utility, then it will perceive seeking input and oversight from **H** as optimal. We characterize **R**'s incentive to seek oversight in this model and show that it depends on a tradeoff between uncertainty about utility evaluations and **H**'s likelihood of making sub-optimal choices.

This suggests that the coffee-making robot above should be designed in order to treat its proxy objective as an *observation* about the intended goal. However, this does not tell us what observation model to use. In Section 4.2, we propose *inverse reward design* (IRD), an observation model for proxy metrics that depends on the environment a proxy was developed for. It assumes that the probability **R** observes a given proxy objective depends on the true utility of the behavior it specifies in the development environment. Thus, utility functions that create incentives for similar behavior in the development environment receive similar weight in the IRD posterior.

This introduces context dependence into **R**'s belief about utility. The posterior distribution that IRD induces over utility evaluations creates high uncertainty about situations where **R** can modify features in novel ways or break correlations that existed in the development environment. In the coffee-making example, **H** writes down a proxy metric that deals primarily with the taste, cost, and caffeine content of different drinks. They evaluate this proxy by running it with a restricted set of options that only includes legal purchases. When interpreted literally, this creates an incentive for **R** to find black market producers to cut costs. However, because black market producers were not an option in the development environment, the IRD posterior has high uncertainty about this course of action, making this choice less attractive.

We use IRD to implement two proxy optimization protocols, inspired by the theoretical results from Section 3.3. First, we use the IRD posterior to design a risk-averse trajectory

optimization method. We show that this can optimize trajectories in novel environments despite the opportunity to cause negative side effects or otherwise game the proxy metric. Next, in Section 4.3, we use IRD to implement a dynamic incentives protocol where **H** provides a sequence of proxy objectives in response to different environments. We present the results of a human subjects study to show that this reduces the cognitive load on participants while generalizing effectively to novel situations.

1.5 The promise and perils of pedagogy

The final component of this thesis argues that the class of *assistance games* identifies a distinct class of alignment solutions called *pedagogic equilibria*. In Chapter 5 we introduce *cooperative inverse reinforcement learning* (CIRL) to formalize assistance games in the context of a fixed objective and a fully-informed principal. This models the *human* actor's incentive to be informative about their goal. Optimal solutions to CIRL games specify a pair of policies: a teaching policy for **H**, $\pi_{\mathbf{H}}$, and a learning policy for **R**, $\pi_{\mathbf{R}}$.

To motivate this change, let us return to the coffee-making robot. Consider the situation where **H** is using the machine for the first time and is faced with a choice: tell **R** which drink to make today or spend 5 minutes to use a reward design interface to give **R** a proxy metric for what types of coffee to make. Unless **H** considers the future value **R** can provide with better information, **H** has no incentive to design a proxy. This means that the problem can no longer be modelled as a POMDP from **R**'s perspective. In a POMDP, the observation distribution is only a function of the world state. However, the incentive for **H** to communicate with **R** depends on **R**'s information state in addition to the world state — there *is* no benefit to purely informative actions if **R** already knows, e.g., what type of beverage to make.

In optimal solutions to CIRL games, $\pi_{\mathbf{H}}$ will deviate from the assumptions made in traditional preference learning to allow the team to get higher total utility. Section 5.1 considers an imitation-learning scenario and shows that **H** can improve **R**'s performance by deviating from the reward-optimal trajectory to steer **R**'s belief to place more weight on the true utility function. Furthermore, this comes with minimal increase in computational cost, in comparison to the POMDP models considered in previous chapters. Section 5.2 shows how to modify the Bellman equation for POMDPs, the mathematical basis on which most POMDP algorithms are developed, so that it accounts for strategic behavior for **H**. We use this to adapt POMDP algorithms to CIRL and develop efficient exact and approximate planning algorithms for CIRL.

Next, Chapter 6 considers the problem of problem of strategy robustness in assistance games. First, Section 6.1 uses the theoretical tools developed in Chapter 5 to compare performance when **R** assumes that **H** behaves pedagogically, but **H** plays a different strategy. It considers a food preparation domain, ChefWorld, where **H**'s goal encodes the desired recipe to cook. In this case, the best response to a pedagogical **H** leads to high utility, even when **H** only maximizes reward in a way that disregards **R**. This is because the pedagogical **H** still pays a lot of attention to immediate reward: the pedagogical incentive essentially breaks ties

between immediate reward-maximizing policies. This means that **R**'s strategy can leverage pedagogy in a way that is robust to certain types of strategy mismatch.

However, when **H** has the capacity to take purely communicative actions (i.e., actions with no direct impact on utility), this creates an opportunity for misalignment caused by the existence of multiple pedagogic equilibria. In Section 6.2, we consider this in the context of a learning principal. More specifically, we consider the case where **H** and **R** are collectively solving *multi-armed bandit* (MAB) [53] problems. Initially, neither **H** nor **R** knows the expected utility from the different arms of the bandit (i.e., **H** does not know the utility associated with each action). Instead, **H** can learn about utility over time from trial and error. **R** can observe **H**'s behavior, but does not directly observe the rewards produced over time.

In the coffee-making example, this models the case where **H** is initially uncertain about what type of coffee they like. In order to maximize utility, they need to experiment with several different options to identify the best one. If **R** believes **H** already knows what they like, then it will misinterpret this exploration and obtain an incorrect preference model. Thus, pedagogical solutions that assume a fully-informed principal are not robust to this type of model mismatch.

We discuss two types of solutions to this *assistive*-MAB problem. In one class of solutions, **H** takes actions that maximize utility in hindsight (i.e., selects actions that greedily maximize immediate expected utility) and **R** ensures that **H** explores enough to eventually identify the optimal arm. In this way, a single robot strategy can assist a wide variety of of human strategies. However, this ignores the potential for pedagogical behavior from **H**.

In the second class of solutions, **H** takes actions that explicitly reveal information about reward observations. In the case of Beta-Bernoulli bandits, **H** can fully convey their reward observations to **R** with, e.g., the win-stay-lose-shift policy, which selects the most recent arm if the associated reward was 1 and a random arm otherwise [126]. This lets the team, $\mathbf{R} \circ \mathbf{H}$, match the performance of the optimal learning strategy for a single agent. Thus, these pedagogic solutions can be utility maximizing.

However, there are multiple pedagogic equilibria and mismatch can lead $\mathbf{R} \circ \mathbf{H}$ to be utility *minimizing*. In this model, **H**'s actions are always unit cost — the only way they impact the world is through **R**'s actions. If **R**'s only assumption about $\pi_{\mathbf{H}}$ is that it is communicative, there is nothing to distinguish win-stay-lose-shift from win-shift-lose-stay. This exposes an important direction for theoretical research on alignment: can we identify alignment solutions that take advantage of pedagogic behavior when it is present, but still provide strong guarantees about team performance when it is not? Our results suggest that alignment problems where **H**'s actions have, at least occasionally, a direct impact on utility produced may admit more robust pedagogical solutions.

1.6 Overview

The rest of this dissertation is organized as follows.

- Chapter 2 goes over the technical background used in the work. We present an overview of sequential decision making methods that covers Markov decision processes (MDP), partially-observed MDPs (POMDP), and inverse reinforcement learning (IRL).

- In Chapter 3, we make the case that principal–agent misalignment is a problem for autonomous systems. We will present a mathematical model of human–robot interaction and show the impact of model misspecification on the performance of the joint system $\mathbf{R} \circ \mathbf{H}$. We then show a general negative result about the ability to optimize for incomplete specifications of utility: in the presence of diminishing returns and shared resources, any incompleteness (modelled as missing attributes of utility) causes $\mathbf{R}$ to eventually drive missing attributes of utility to 0. We show that this changes if $\mathbf{R}$'s incentives are modified to include weak dependence on all attributes so it can minimize impact or if incentives can be dynamically updated as the state of the world changes.

- In Chapter 4, we consider the role that uncertainty about the goal (represented as uncertainty about the mapping from states to utility) plays in optimal assistance behaviors. We show that uncertainty creates generic incentives to seek supervision. We define the problem of *inverse reward design*, where the goal is to determine a distribution of potential 'intended' goals based on an observation of a proxy reward and a development environment. We show how this machinery can be used to implement the impact avoidance and dynamic-incentive protocols from the previous chapter.

- In Chapter 5, we examine the role that pedagogy on the part of $\mathbf{H}$ plays in increasing the value of the interaction $\mathbf{R} \circ \mathbf{H}$. We show that the optimal strategy for $\mathbf{H}$ accounts for the informational value of their actions to $\mathbf{R}$. As a result, we generalize the assistance POMDPs from the previous chapters to *assistance games* that model strategic behavior from $\mathbf{H}$. We introduce *cooperative inverse reinforcement learning* (CIRL) to formalize optimal assistance for a fully informed, strategic, principal with a static utility function. We consider the problem of computing pedagogic equilibria in CIRL games. We show that these equilibria increase utility for $\mathbf{H}$ and can be computed efficiently with modified POMDP solution algorithms.

- Chapter 6 considers the problem of strategy robustness in pedagogic solutions to alignment problems. First, it shows that pedagogic solutions that are also sensitive to the immediate reward can perform well, even when $\mathbf{H}$ does not attempt to coordinate with $\mathbf{R}$. Then, it considers an assistive variant of the classic multi-armed bandit problem. This shows the potential robustness of reward-maximizing strategies as a single policy for $\mathbf{R}$ can successfully assist a broad class of $\mathbf{H}$ strategies. It also highlights the potential for high-utility solutions that rely on $\mathbf{H}$ to encode their reward observations into their behavior. This also introduces a failure mode from equilibrium mismatch where $\mathbf{R}$ misinterprets $\mathbf{H}$'s actions.

- Chapter 7 collects related work.

- Chapter 8 concludes with a short discussion of directions for further research on value alignment.

Chapter 2

Preliminaries

We begin by going over the background material that assistance games build on. Our work will attempt to build simple extensions to standard formulations for decision-making in artificial intelligence that model the principal–agent alignment problems that arise. More specifically, we will present a formulation of optimal assistance as a sequential interaction between a *principal*, the human **H**, and an *agent*, the robot **R** where

1. **H** and **R** have a shared objective, maximizing utility for **H**;
2. **H** knows the objective, but **R** does not; and
3. **R** can learn about the objective by observing **H**'s actions.

We present this model, *cooperative inverse reinforcement learning* (CIRL), formally in Chapter 5. It extends a *Markov decision process* (MDP) to introduce uncertainty about a static, asymmetrically observed objective and another actor, **H**. In this section, we give a brief overview of the background for our method: MDPs; their partially observed extension, *partially-observed Markov decision processes* (POMDP); and the problem of inferring objectives from optimal behavior, *inverse reinforcement learning* (IRL).

2.1 Markov Decision Process

Markov decision processes (MDP) are the standard formulation of optimal sequential decision-making in artificial intelligence. MDPs model the world through a *state* s that captures the relevant aspects of the environment needed to measure utility $\mathcal{U}(s)$ and predict the distribution the next state, conditioned on the current action $T\left(s^{(t)}, a^{(t)}\right)$. A solution to an MDP is a *policy* $\pi(s)$ that determines a mapping from states to actions. An optimal policy maximizes the expected discounted sum of utilities.

Our formulation of MDPs for this work will differ slightly from standard formulations. We will be interested in extensions of MDPs where the state is partially observed and encodes

information about the goal. As a result, we will take care to distinguish the *environment*, the model of the physical world, and the *goal*, the specification of optimal behavior. This distinguishes *objective* properties of the physical world from the *normative* properties that define optimal behavior. The environment variables are *identifiable* — in the limit of infinite data, these variables can be learned — while the goal is *not*, without additional assumptions about the relationship between goals and behavior.

We begin by defining a Markov environment.

Definition 1. (Markov Environment)

An environment is a tuple

$$E = \langle \{\mathcal{S}, P_{\mathcal{S}}\}, \mathcal{A}, T \rangle :$$

$\mathcal{S}$ *A set of world states.* $s \in \mathcal{S}$;

$P_{\mathcal{S}}$ *A distribution over the initial state of the world.* $P_{\mathcal{S}} \in \Delta(\mathcal{S})$;

$\mathcal{A}$ *A set of actions.* $a \in \mathcal{A}$;

T *A transition distribution that determines the distribution over next states, given the previous state and action.* $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$;

Unless otherwise specified, any environment we consider is a Markov environment. An environment defines the space of trajectory distributions that an agent can execute by changing its policy. The optimal trajectory distribution, and hence the optimal policy, is specified by a utility function $\mathcal{U}$ that determines a total ordering over states. We will consider utility functions that depend on the current state and action $\mathcal{U}(s, a)$ and refer to the combination of utility function and discount factor as a *utility model*.

Definition 2. (Utility Model)
For a given environment $E = \langle \{\mathcal{S}, P_{\mathcal{S}}\}, \mathcal{A}, T \rangle$*, an associated utility model is a pair*

$$U = \langle \mathcal{U}, \gamma \rangle :$$

$\mathcal{U}$ *A utility function that specifies a total ordering over state-action pairs* $(s, a) \in \mathcal{S} \times \mathcal{A}$ *from* E, $\mathcal{U} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$;

γ *A discount factor that trades off between current and future utility,* $\gamma \in [0, 1)$.

An agent behaving optimally according to U will choose actions to maximize the discounted sum of future utility:

$$\max_{\vec{a}} \quad \mathbb{E}\left[\sum_t \gamma^t \mathcal{U}\left(s^{(t)}, a^{(t)}\right) \middle| s^{(t+1)} \sim T\left(s^{(t)}, a^{(t)}\right) \right] \tag{2.1}$$

It is typical in MDP research to use r to represent this reward function and reserve utility for the sum of rewards across time. In this work, we will use utility $\mathcal{U}$ to avoid confusion with the robot player $\mathbf{R}$ in two-player assistance games. Together, Definition 1 and Definition 2 allow us to define a Markov decision process as a pair consisting of a Markov environment and a utility model.

Definition 3. (Markov Decision Process)
*A Markov decision process (*MDP*) is a tuple*

$$M = \langle E, U \rangle :$$

- E *A Markov environment* $\langle \{\mathcal{S}, P_{\mathcal{S}}\}, \mathcal{A}, T \rangle$ *that specifies a set of world states* $s \in \mathcal{S}$*, the initial state distribution* $P_{\mathcal{S}} \in \Delta(\mathcal{S})$*, a set of actions* $a \in \mathcal{A}$*, and a transition distribution over next state conditioned on previous state and action* $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$*;*
- U *A utility model* $\langle \mathcal{U}, \gamma \rangle$ *that specifies a total ordering over state-action pairs* $\mathcal{U} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *and a discount factor* $\gamma \in [0, 1)$*.*

In an MDP, the goal is to identify a mapping from states to (a distribution over) actions so that the resulting Markov chain optimizes the expected sum of future utility. This mapping is called a *policy*, $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. The *value* of a state s under policy π is the expected sum of discounted utilities obtained starting in s and following π:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_t \gamma^t \mathcal{U}\left(s^{(t)}, a^{(t)}\right) \middle| s^{(t+1)} \sim T\left(s^{(t)}, a^{(t)}\right); a^{(t)} \sim \pi\left(s^{(t)}\right); s^{(0)} \sim P_{\mathcal{S}} \right]. \tag{2.2}$$

The *action-value* $Q^{\pi}(s, a)$ of action a in state s under policy π is the value obtained by executing a initially and following π thereafter. It satisfies a dynamic programming relation with V:

$$Q^{\pi}(s, a) = \mathcal{U}(s, a) + \gamma \sum_{s'} V^{\pi}(s') T(s' | s, a). \tag{2.3}$$

We also have that $V^{\pi}(s) = \mathbb{E}\left[Q^{\pi}(s, a) | a \sim \pi(s) \right]$. The optimal policy π^* maximizes V^{π}.

$$\pi^* \in \arg\max_{\pi} \mathbb{E}\left[V^{\pi} | s^{(0)} \sim P_{\mathcal{S}} \right]. \tag{2.4}$$

The value function and action-value function of π^* are simply written V and Q. They obey the *Bellman* dynamic programming equations:

$$V(s) = \max_{a \in \mathcal{A}} Q(s, a). \tag{2.5}$$

There are several efficient *planning* algorithms to compute optimal policies for MDPs. The most direct approach is to initialize a vector of values and alternate between updating Q

based on $V$and applying Equation 2.5 to update V until the vector of values converges. Then the optimal policy can be recovered from Q:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q(s, a). \tag{2.6}$$

It is often useful to discus the sequence of states that an actor has traversed. This is the *state-action* history τ.

Definition 4. ((State-Action History))
An state-action history τ, is a sequence of state, action pairs

$$\tau = (s, a)^{\{1:t\}} \in (\mathcal{S} \times \mathcal{A})^*. \tag{2.7}$$

A state-action history is often called a *trajectory* in robotics applications. We will use both terms interchangeably. We will use $\tau^{(t)}$ to refer to the t^{th} state-action pair $\left(s^{(t)}, a^{(t)}\right)$.

2.2 Partially-Observed Markov Decision Process

A *partially-observed Markov decision process* (POMDP) is an extension of MDPs to account for unknown aspects of the world state. A POMDP is defined from a reference MDP. However, instead of directly observing the world state s, the agent sees an observation $o \in \mathcal{O}$ at random from an *observation distribution* that is conditioned on the current state: $P_{\mathcal{O}|\mathcal{S}}(s) \in \Delta(\mathcal{O})$.

Definition 5. (Observation Model)
For a given state space $\mathcal{S}$, the associated observation model is a tuple $O = \langle \mathcal{O}, P_{\mathcal{O}|\mathcal{S}} \rangle$:

$\mathcal{O}$ *A set of* observations*, $o \in \mathcal{O}$;*

$P_{\mathcal{O}|\mathcal{S}}$ *A distribution over observations conditioned on the current state, $P_{\mathcal{O}|\mathcal{S}}(s) \in \Delta(\mathcal{O})$.*

Then, we can define a POMDP as a tuple of an environment, an observation model, and a utility model.

Definition 6. (Partially-Observed Markov Decision Process)
A POMDP *M augments an* MDP *$\langle E, U \rangle$, with an observation model $\langle \mathcal{O}, P_{\mathcal{O}|\mathcal{S}} \rangle$. Formally, it is a tuple of an environment, an observation model, and a utility model*

$$M = \langle E, O, U \rangle :$$

E *a Markov environment $\langle \{\mathcal{S}, P_{\mathcal{S}}\}, \mathcal{A}, T \rangle$ that specifies a set of world states $s \in \mathcal{S}$, the initial state distribution $P_{\mathcal{S}} \in \Delta(\mathcal{S})$, a set of actions $a \in \mathcal{A}$, and a transition distribution over next state conditioned on the previous state-observation-action tuple $T : \mathcal{S} \times \mathcal{O} \times \mathcal{A} \to \Delta(\mathcal{S})$;*

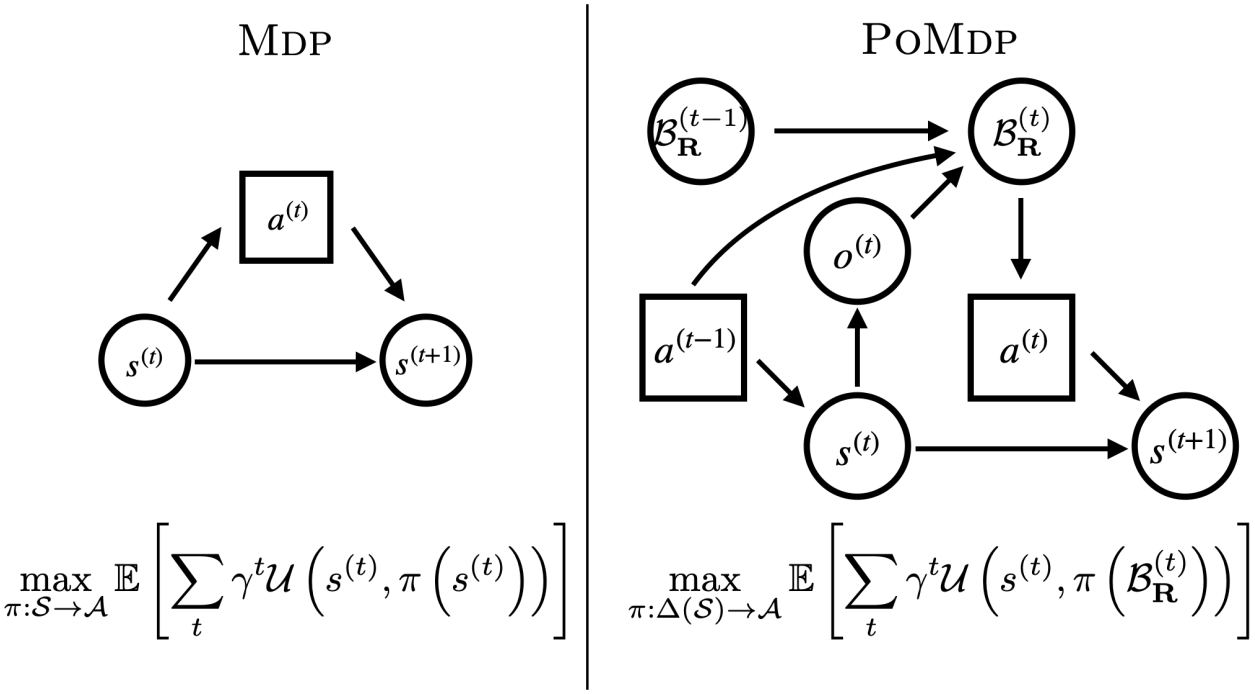


Figure 2.1: An influence diagram illustrating the conditional relationships between random variables nodes (illustrated with circles) and decision nodes (illustrated with squares) in sequential decision making. **Left:** Markov decision processes (MDP) model situations where the system is able to take action directly in response to world state. **Right:** A partially-observable Markov decision processes (POMDP) model decisions that must be made based on partial observations of the world state. In this case the actor, the robot $\mathbf{R}$, acts based on a belief state that summarizes the history of actions and observations.

- O *an observation model* $\langle \mathcal{O}, P_{\mathcal{O}|\mathcal{S}} \rangle$ *that specifies a set of observations* $o \in \mathcal{O}$ *and a distribution on observations conditioned on state* $P_{\mathcal{O}|\mathcal{S}}(s) \in \Delta(\mathcal{O})$*; and*
- U *a utility model* $\langle \mathcal{U}, \gamma \rangle$ *that specifies a total ordering over state-action pairs* $\mathcal{U} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *and a discount factor* $\gamma \in [0, 1)$.

Note that we have extended the transition distribution to also depend on the previous observation $o^{(t)}$ in addition to the state-action pair $\left(s^{(t)}, a^{(t)}\right)$. This will simplify notation later on when the observations will correspond to actions taken by the human actor $\mathbf{H}$.

The primary change is in the domain of the policy. In an MDP, a policy maps a state $s^{(t)}$ into the next action $a^{(t)}$. In a POMDP, $s^{(t)}$ is unobserved, so the policy cannot directly depend on it. Instead, $\mathbf{R}$ chooses actions based on its *observations* $o^{(t)} \sim P_{\mathcal{O}|\mathcal{S}}\left(s^{(t)}\right)$. It is generally suboptimal for a policy to depend on $o^{(t)}$ alone. An optimal policy will take advantage of all the information available. As a result, work on POMDPs generally considers

policies that depend on the full history of an agent's interaction with the environment. This brings in a dependence on the previous actions through the transition distribution T.

Definition 7. (Observation-Action History)
An observation-action history τ is a tuple of sequence of observation and a sequence of actions

$$\tau = \left(o^{\{1:t\}}, a^{\{1:t\}}\right) \in (\mathcal{O} \times \mathcal{A})^* . \tag{2.8}$$

We use τ to denote these action-observation histories to maintain consistency with the state-action histories (i.e., trajectories) in Section 2.1.

A policy is now a mapping from the current observation $o^{(t)}$ and one such sequence to the next action $a^{(t)}$,

$$\pi : \mathcal{O} \times (\mathcal{O} \times \mathcal{A})^* \to \mathcal{A}. \tag{2.9}$$

In discrete environments, this leads to an exponential growth in the size of the domain for an optimal policy. As a result, POMDPs are hard. While an MDP can be solved efficiently by a variety of methods, POMDPs do not admit a polynomial time algorithm.

Most approaches to solving POMDPs rely on the concept of a *belief* state (also called an *information* state).

Definition 8. (Belief State)
R*'s belief state at time t is the posterior distribution of states, given the action-observation history and the current observation*

$$\mathcal{B}_{\mathbf{R}}^{(t)} = P\left(s^{(t)} \left| o^{\{1:t\}}, a^{\{1:t-1\}} \right.\right) \in \Delta\left(\mathcal{S}\right). \tag{2.10}$$

This distribution is analogous to the filtering distribution in hidden Markov models. It can also be defined recursively as

$$\mathcal{B}_{\mathbf{R}}^{(t)}\left(s^{(t)}\right) \propto P_{\mathcal{O}|\mathcal{S}}\left(o^{(t)} \left| s^{(t)} \right.\right) \sum_{s^{(t-1)} \in \mathcal{S}} T\left(s^{(t)} \left| s^{(t-1)}, o^{(t-1)}, a^{(t-1)} \right.\right) \mathcal{B}_{\mathbf{R}}^{(t-1)}\left(s^{(t-1)}\right). \tag{2.11}$$

A classic result in POMDP theory is that the optimal policy only depends on the observation-action history through **R**'s belief state [149, 81]. Thus, work on POMDPs typically restricts its attention to policies that map the robot's belief $\mathcal{B}_{\mathbf{R}}^{(t)}$ into the next action $a^{(t)}$,

$$\pi : \Delta\left(\mathcal{S}\right) \to \mathcal{A}. \tag{2.12}$$

Thus, the optimal solution to a POMDP satisfies

$$\pi^* \in \arg\max_{\pi} \quad \mathbb{E}\left[\sum_t \gamma^t \mathcal{U}\left(s^{(t)}\right) \middle| s^{(t+1)} \sim T\left(s^{(t)}, o^{(t)}, \pi\left(\mathcal{B}_{\mathbf{R}}^{(t)}\right)\right)\right]. \tag{2.13}$$

POMDPs are more computationally challenging than MDPs. MDPs can be solved in polynomial time by a variety of methods. POMDPs, on the other hand, are PSPACE-complete [18].

Speaking loosely, this complexity arises from the exponential growth in the policy space compared with MDPs. Policies in POMDPs are often represented as *conditional plans*. A conditional plan σ, is a tree of local decision rules v that map the most recent observation into an action $v(o) \in \Delta(\mathcal{A})$. The number of conditional plans grows exponentially in the horizon, and this accounts for the computational complexity of POMDPs.

Instead of tracking a value for each state, dynamic programming algorithms for POMDPs track a vector of values called an α-vector $\alpha \in \mathbb{R}^{|\mathcal{S}|}$. For a conditional plan σ, $\alpha^\sigma(s)$ contains the value of following σ from s. The value of a given belief state $\mathcal{B}_{\mathbf{R}}$ can be computed with an inner product

$$V^\sigma(\mathcal{B}_{\mathbf{R}}) = \sum_{s \in \mathcal{S}} \mathcal{B}_{\mathbf{R}}(s) \cdot \alpha^\sigma(s). \tag{2.14}$$

Exact POMDP algorithms typically maintain a set of conditional plans and associated α^σ. They alternate between generating a new set of conditional plans by prepending new decision rules on to the existing conditional plans, computing the associate α^σ, and then pruning *dominated* conditional plans.

2.3 Inverse Reinforcement Learning

In this work, we are interested in the ability of autonomous systems to learning about their objectives and adjust their behavior accordingly. The standard formulation of this problem in artificial intelligence is *inverse reinforcement learning* (IRL) [115] or *inverse optimal control* [83]. If the objective of planning is to determine optimal behavior for a given utility function, the objective of IRL is the opposite: given observations of optimal behavior, identify the utility function it maximizes. We will focus on Bayesian formulations of IRL, where the goal is to infer a distribution on $\mathcal{U}$, given observations of optimal behavior for a particular MDP.

The actor being observed in IRL is typically referred to as the 'expert' or 'demonstrator.' In our application, this actor is the principal in an assistance problem, $\mathbf{H}$. To maintain consistency with that application we will we use $\pi^*_{\mathbf{H}}$ to denote this expert policy. Formally, we represent $\mathbf{H}$'s preferences with a new state variable, $\mathbf{H}$'s *type* $\theta \in \Theta$. We extend the utility function accordingly.

Definition 9. (Parameterized Utility Model)
For a given environment $E = \langle \{\mathcal{S}, P_{\mathcal{S}}\}, \mathcal{A}, T \rangle$ and type space Θ, an associated parameterized utility model is a pair

$$U = \langle \mathcal{U}_\theta, \gamma \rangle :$$

- $\mathcal{U}_\theta$ *A parameterized utility function that specifies a total ordering over state-action pairs $(s, a) \in \mathcal{S} \times \mathcal{A}$ from E for each type $\theta \in \Theta$, $\mathcal{U}_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$;*
- γ *A discount factor that trades off between current and future utility, $\gamma \in [0, 1)$.*

In future sections, we will formalize principal–agent alignment problems as PoMDPs that build on IRL. As a result, we will treat θ as an unobserved part of the state. Depending on the situation, we will interchangeably denote θ as a subscript of $\mathcal{U}$ or an argument:

$$\mathcal{U}_\theta(s,a) = \mathcal{U}(s,a;\theta).$$

This work largely builds on Bayesian approaches to IRL, which adopt a prior distribution over types, $P_\theta \in \Delta(\Theta)$. Most Bayesian IRL methods use the behavior model introduced in Ramachadran and Amir [122], where actions are taken in proportion to the exponential of their Q-value,

$$\pi^\beta_{\mathbf{H}}(a|s;\theta) \propto \exp(\beta Q(s,a;\theta)). \tag{2.15}$$

This relaxes the optimality constraints on $\mathbf{H}$ and allows supoptimal actions to be taken with an exponential decrease in probability as the action-value decreases. In this context, β represents how optimal $\mathbf{H}$ is. When $\beta = 0$, $\mathbf{H}$ selects actions uniformly at random. As β increases, $\mathbf{H}$ becomes more likely to select the optimal action. $\pi^*_{\mathbf{H}}$ is recovered in the limit as $\beta \to \infty$;

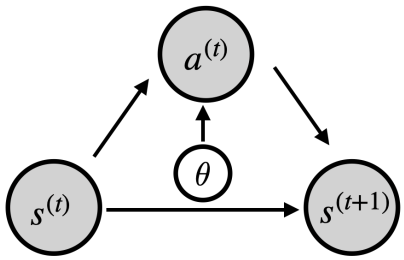
$$\lim_{\beta\to\infty} \pi^\beta_{\mathbf{H}}(s) = \pi^*_{\mathbf{H}}(s) \propto \begin{cases} 1 & a \in \arg\max_a Q(s,a;\theta) \\ 0 & \textbf{else} \end{cases}. \tag{2.16}$$

Readers familiar with IRL should note that, in this formulation, the behavioral model is related to, but distinct from, the expert (or target) policy — as the term is typically used in the IRL literature. The expert policy usually refers the behavior being demonstrated (e.g., the aerobatics tricks in Abbeel, Coates, and Ng [1]) and is represented as a policy in the original MDP (i.e., the expert policy is a mapping $\mathcal{S} \to \mathcal{A}$). $\pi_{\mathbf{H}}$ is technically a *meta-policy* with respect to the original MDP. It depends on the current state and $\mathbf{H}$'s goal, as indicated by their type. The expert policy can be recovered from $\pi_{\mathbf{H}}$ by fixing a type: $\pi_{\mathbf{H}}(\cdot;\theta)$. We will refer to the combination of a type space Θ, prior P_θ, and behavioral model $\pi_{\mathbf{H}}$, as a *population model*.

Definition 10. (Population Model)
A population model is a tuple $\langle \Theta, P_\theta, \pi_{\mathbf{H}} \rangle$*:*

- Θ *A space of types,* $\theta \in \Theta$;
- P_θ *A distribution over* Θ*,* $P_\theta \in \Delta(\Theta)$;
- $\pi_{\mathbf{H}}$ *A mapping from states* s *and types* θ *to a distribution over actions,* $\pi_{\mathbf{H}} : \mathcal{S} \times \Theta \to \Delta(\mathcal{A})$.

Together this allows us to formalize a Bayesian inverse reinforcement learning (IRL) problem.



$$\pi_{\mathbf{H}} \in \arg\max_{\pi} \mathbb{E}\left[\sum_{t} \gamma^{t} \mathcal{U}\left(s^{(t)}, \pi\left(s^{(t)}\right); \theta\right)\right]$$

Figure 2.2: An illustration of the *inverse reinforcement learning* problem as a Bayesian network. The goal is to infer the utility function, represented by a generic type variable, θ, given observations of actions and context in which they are taken.

Definition 11. (Bayesian Inverse Reinforcement Learning)
*A Bayesian inverse reinforcement learning (*IRL*) problem is defined by an environment* $E = \langle \{\mathcal{S}, P_{\mathcal{S}}\}, \mathcal{A}, T \rangle$*, a population model* $\Pi = \langle \Theta, P_\theta, \pi_{\mathbf{H}} \rangle$*, a parameterized utility model* $U = \langle \mathcal{U}_\theta, \gamma \rangle$*, and an expert trajectory (i.e., state-action history)* $\tau_{\mathbf{H}}$*. Formally, this is a tuple*

$$\langle E, \Pi, U, \tau_{\mathbf{H}} \rangle :$$

- E *a Markov environment* $\langle \{\mathcal{S}, P_{\mathcal{S}}\}, \mathcal{A}, T \rangle$ *that specifies a set of world states* $s \in \mathcal{S}$*, the initial state distribution* $P_{\mathcal{S}} \in \Delta(\mathcal{S})$*, a set of actions* $a \in \mathcal{A}$*, and a transition distribution over next state conditioned on the previous state-action pair* $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$;
- Π *a population model that specifies a space of types* $\theta \in \Theta$*, a distribution over types* $P_\theta \in \Delta(\Theta)$*, and a behavioral model that specifies a distribution over actions conditioned on state-type tuples* $\pi_{\mathbf{H}} : \mathcal{S} \times \Theta \to \Delta(\mathcal{A})$*; and*
- U *a utility model* $\langle \mathcal{U}, \gamma \rangle$ *that specifies a total ordering over state-action pairs* $\mathcal{U} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *and a discount factor* $\gamma \in [0, 1)$.

The goal of Bayesian IRL is to recover $\mathbf{H}$'s type θ from the observed trajectory $\tau_{\mathbf{H}}$ of state-action pairs. Specifically, a solution to a Bayesian IRL problem is an algorithm that computes the posterior distribution on types given a state-action history. In keeping with the previous section, we will denote this target distribution as $\mathcal{B}_{\mathbf{R}}$.

$$\mathcal{B}_{\mathbf{R}}(\theta \,|\tau_{\mathbf{H}}) = P\left(\theta \,\middle|\, \left(s^{(t)}, a^{(t)}\right); a^{(t)} \sim \pi_{\mathbf{H}}\left(s^{(t)}; \theta\right)\right) \tag{2.17}$$

$$\propto \prod_{\left(s^{(t)}, a^{(t)}\right) \in \tau_{\mathbf{H}}} \pi_{\mathbf{H}}\left(a^{(t)} \,\middle|\, s^{(t)}; \theta\right). \tag{2.18}$$

Figure 2.2 illustrates the conditional dependencies for IRL as a Bayesian network. One of the central challenges in designing algorithm for IRL is that the space of possible trajectories is quite large. In order to run inference, most algorithms will need to identify, implicitly or explicitly, what the behavior *would* have been for different candidate objectives. This can be done naively by replanning for a wide range of candidates, but that is often unacceptably slow. As a result, many approaches to IRL are built around techniques that manage this compute cost.

Ramachadran and Amir [122] leverage locality in the sequence of reward functions considered to speed up re-planning at successive steps of a Markov-chain Monte Carlo [23] method. Their central observation is that dynamic programming solutions can be re-used efficiently if the changes in candidate reward functions are sparse.

A special case of interest is that of *linear* reward functions. That is, reward functions that can be decomposed into a set of weights $w \in \mathbb{R}^K$ and feature function $\phi : \mathcal{S} \to \mathbb{R}^K$ such that $\mathcal{U}_\theta(s) = w^\top \phi(s)$. We will use w to represent $\mathbf{H}$'s type in this case, instead of the more general θ. Abbeel and Ng [2] showed that matching the experts reward in this case can be reduced to matching the feature expectations from the demonstrations. Syed and Schapire [162] takes this idea further and shows how to use adversarial assumptions to *improve* on the demonstrator's performance. We will use this max-min idea to implement our risk-averse trajectory optimization method in Section 4.2.

Chapter 3

Misalignment

Almost any autonomous agent relies on two key components: a specified goal or reward function for the system and an optimization algorithm to compute the optimal behavior for that goal. This procedure is intended to produce value for a *principal*: the user, system designer, or company on whose behalf the agent acts. Research in AI typically seeks to identify more effective optimization techniques under the, often unstated, assumption that better optimization will produce more value for the principal.

If the specified objective is a complete and accurate representation of the principal's goals, then this assumption is surely justified. However, the specified goal for a system is often an incomplete or inaccurate representation of the intended goal. We have ample evidence from both theory and application, that it is impractical, if not impossible, to provide a complete specification of preferences to an autonomous agent. The gap between specified proxy rewards and the true objective creates a principal–agent problem between the designers of an AI system and the system itself: the objective of the principal (i.e., the designer) is different from, and thus potentially in conflict with, the objective of the autonomous agent. In human principal–agent problems, seemingly inconsequential changes to an agent's incentives often lead to surprising, counter-intuitive, and counter-productive behavior [85]. Consequently, we must ask when this *misalignment* is costly: When is it counter-productive to optimize for an incomplete proxy? What properties of a decision problem lead to overoptimization?

In this chapter, we make the case that principal–agent costly misalignment is to be expected when autonomous systems optimize for incomplete measurements of value. We start by analyzing the problem as a type of POMDP, where the principal, **H** is represented by an observation model that provides observations about the reward associated with different actions. We call this a *supervision*-POMDP, as **H**, in effect, provides direct supervision to the (robot) agent, which we denote with **R**. In the next chapters, we will build on this model to consider *assistance*-POMDPs, where **H**'s actions can also modify the state of the world, leading up to our game-theoretic model of assistance, *cooperative inverse reinforcement learning* (CIRL), where **H** has the capacity to behave strategically in response to **R**'s behavior.

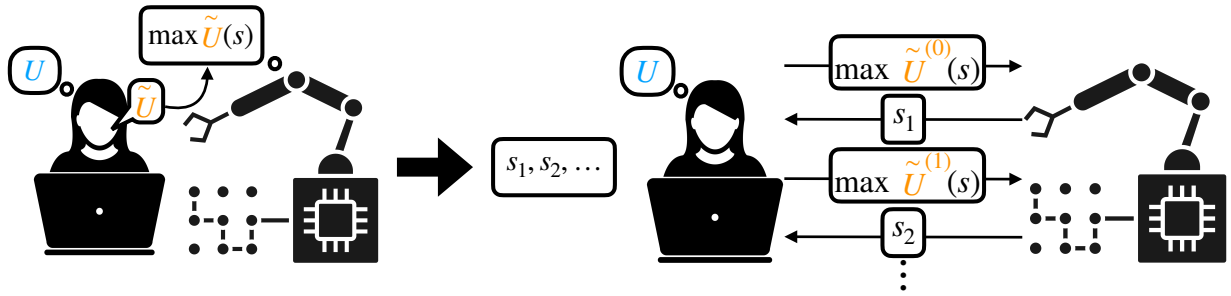


Figure 3.1: A comparison of our two models for reward design. In a static paradigm (**left**), the designer provides a single proxy objective $\tilde{\mathcal{U}}$. Theorem 6 shows that this leads to very suboptimal outcomes if $\tilde{\mathcal{U}}$ does not have the same support as $\mathcal{U}$. Theorem 10 shows that the dynamic incentives paradigm (**right**), where the designer specifies a sequence of objectives $\{\tilde{\mathcal{U}}^{(1)}, \tilde{\mathcal{U}}^{(2)}, \tilde{\mathcal{U}}^{(3)}, \ldots\}$ in response to incremental changes, *does* reliably increase utility.

We use this model to illustrate two results. First, in the optimal configuration of $\mathbf{R} \circ \mathbf{H}$, $\mathbf{R}$ learns from $\mathbf{H}$'s past actions to identify their objective, represented with w. This leads $\mathbf{R}$ to, eventually, take actions that differ from $\mathbf{H}$'s selection. This allows $\mathbf{R} \circ \mathbf{H}$ to produce more utility than $\mathbf{H}$ would alone. Second, consider the problem of misspecification. We show that missing features lead $\mathbf{R} \circ \mathbf{H}$ to be *inconsistent*: even in the limit of infinite data $\mathbf{R} \circ \mathbf{H}$ takes suboptimal actions with finite probabilities.

Section 3.2 builds on this insight to show a general negative result about the ability to optimize for incomplete specifications of utility. We consider an interaction where $\mathbf{H}$ only has one action available: write down a proxy objective. We prove that, in the presence of diminishing returns and shared resources, any incompleteness (modelled as missing features of utility) causes $\mathbf{R}$ to effectively minimize missing attributes of utility. In Section 3.3, we consider mitigation strategies for misalignment. We show that this changes if $\mathbf{R}$'s incentives are modified to include weak dependence on all features so it can minimize impact or if the support of $\mathbf{R}$'s incentives can be dynamically updated as the state of the world changes. Figure 3.1 illustrates the fixed proxy model studied in Section 3.2 and the dynamic incentives protocol introduced in Section 3.3.

3.1 A Supervision POMDP

Should robots be obedient? The reflexive answer to this question is yes. A coffee-making robot that doesn't listen to your coffee order is not likely to sell well. However, the story of King Midas illustrates the complexity to this seemingly simple question: blindly obedient systems may easily place too much faith in their users. A self-driving car should certainly defer to its owner when she tries taking over because it's driving too fast in the snow. The car shouldn't let a child accidentally turn on the manual driving mode. The suggestion that

it might sometimes be better for an autonomous systems to be disobedient is not new [173, 132]. For example, this is the idea behind "Do What I Mean" systems [164] that attempt to act based on the user's intent rather than the user's literal order.

In this section, we use supervision-POMDPs to analyze this tradeoff. We formalize obedience in a supervision-POMDP as the fraction of times that $\mathbf{R}$ copies $\mathbf{H}$'s action. $\mathbf{H}$ and $\mathbf{R}$ are cooperative, but $\mathbf{H}$ knows the reward parameters θ and $\mathbf{R}$ does not. $\mathbf{H}$'s actions are effectively orders. $\mathbf{R}$ can decide whether to obey or not. We first show that it is optimal for $\mathbf{R}$ directly to imitate $\mathbf{H}$ when $\mathbf{H}$ implements an optimal policy π^*. Then, we consider the case of a suboptimal $\mathbf{H}$. We show that if $\mathbf{R}$ tries to infer θ from $\mathbf{H}$'s orders and then acts by optimizing its estimate of θ, then it can always do better than a blindly obedient robot. Thus, forcing $\mathbf{R}$ to be blindly obedient does not come for free: it requires giving up the potential to surpass human performance.

The Supervision POMDP

There's a clear relationship between POMDPs and IRL. Naturally, both models are relevant to principal–agent problems in AI. In IRL, the goal is to infer $\mathbf{H}$'s goal. POMDPs, on the other hand, define the optimal way for $\mathbf{R}$ to respond to uncertainty about the state of the world. Our first formulation of an assistance problem, supervision-POMDPs, integrates IRL-like inference into a POMDP formalism.

The main idea is to treat θ as an unobserved component of the state and $\pi_{\mathbf{H}}$ as the observation model. In a supervision-POMDP $\mathbf{H}$'s actions are purely informative — that is, they do not directly change the state of the world[1]. We represent this as a POMDP where the observation space $\mathcal{O}$ is identical to the action space $\mathcal{A}$. Figure 3.2 shows an illustration of the sequence of events.

Definition 12. (Supervision-POMDP)
Let environment $E = \langle \{\mathcal{S}, P_{\mathcal{S}}\}, \mathcal{A}, T\rangle$, population model $\Pi = \langle \Theta, P_\theta, \pi_{\mathbf{H}}\rangle$, and parameterized utility model $U = \langle \mathcal{U}_\theta, \gamma\rangle$. Define O_Π and E' as follows

$$O_\Pi = \langle \mathcal{S} \times \mathcal{A}, \delta \times \pi_{\mathbf{H}}(s;\theta)\rangle; \tag{3.1}$$

$$E' = \langle \{\mathcal{S} \times \Theta, P_{\mathcal{S}} \times P_\theta\}, \mathcal{A}, T\rangle. \tag{3.2}$$

*The associated supervision-*POMDP *is a* POMDP $M = \langle E', O_\Pi, U\rangle$.

The defining feature of a supervision-POMDP is the combination of shared action and observation spaces with an unobserved type for $\mathbf{H}$, θ, that is revealed through samples from $\pi_{\mathbf{H}}$. The world state s is directly observed and, as a result, we will abuse notation slightly and omit the state component of observations. We denote the t^{th} observation in a supervision-POMDP with $a_{\mathbf{H}}^{(t)}$ and the t^{th} action with $a_{\mathbf{R}}^{(t)}$. We will similarly treat $\mathcal{B}_{\mathbf{R}}$ as a distribution

[1] In Chapter 4, we will present a generalization of supervision-POMDPs where $\mathbf{H}$'s actions provide information and change the world state.

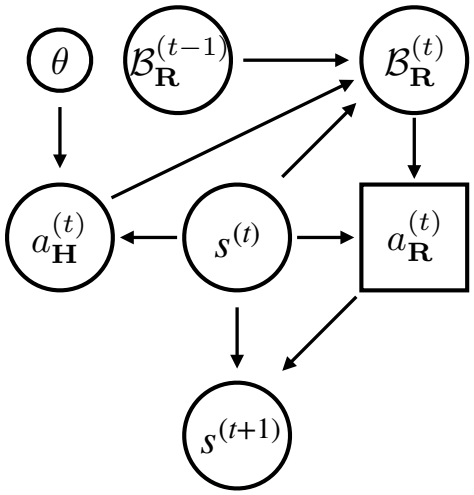


Figure 3.2: An illustration of the conditional dependencies for one time step of a supervision-POMDP. $\mathbf{H}$'s action $a_{\mathbf{H}}^{(t)}$ is selected, conditional on their objective, θ, and the world state $s^{(t)}$. $\mathbf{R}$'s action $a_{\mathbf{R}}$, is selected based on its current belief about θ, $\mathcal{B}_{\mathbf{R}}^{(t)}$, and the world state.

over θ and think of $\pi_{\mathbf{R}}$ as a mapping from tuples of world state and belief. We will treat θ as a parameter of the value function:

$$Q_\theta(s,a) = Q_{\mathbf{H}}(s,a;\theta); V_\theta(s) = V_{\mathbf{H}}(s;\theta). \tag{3.3}$$

We have used $Q_{\mathbf{H}}$ and $V_{\mathbf{H}}$ to emphasize that these describe the expected value of a state(-action pair) in the original MDP. That is, they represent the value function when $\mathbf{H}$ directly controls the system. Thus, V_θ is equivalent to $\alpha^{\pi_{\mathbf{R}}^{\text{TELEOP}}}$, the α for a conditional plan that executes $\pi_{\mathbf{R}}^{\text{TELEOP}}$. The value of a belief state $\mathcal{B}_{\mathbf{R}}$ is $\mathbb{E}\left[V_\theta | \, \theta \sim \mathcal{B}_{\mathbf{R}}\right]$.

We will focus our analysis on a supervision-POMDP with independent transitions. That is, each state is sampled independently from a fixed prior distribution. Each action is associated with a feature vector $\phi_a \in \mathbb{R}^M$. $\mathbf{H}$'s preferences are represented as a vector $\theta \in \mathbb{R}^M$ that determines a linear utility function $\mathcal{U}(s,a;\theta) = \theta^\top \phi_a$. The system state is represented as a matrix where each column encodes the features for the associated action $s = [\phi_a] \in \mathbb{R}^{M \times K}$.

This removes the sequential aspect of the problem and reduces it to a type of modified contextual bandit [98]. As a result, we will call this assistance problem a *contextual* supervision-POMDP.

Definition 13. (Contextual supervision-POMDP)

Let $P_\phi \in \Delta\left(\mathbb{R}^{M \times K}\right)$ be a distribution over features $\phi \in \mathbb{R}^M$. Let $U = \langle \mathcal{U}_\theta, \phi, \gamma \rangle$ be a linear utility model with $\mathcal{U}_\theta(s,a) = \theta^\top \phi(s,a)$. Let $\Pi = \langle \Theta, P_\theta, \pi_{\mathbf{H}} \rangle$ be a population model

with an associated observation model O_Π. *Define the environment*

$$E_\phi = \langle \{\mathbb{R}^{M\times K}, P_\phi\}, [1,\dots,K], P_\phi \rangle \tag{3.4}$$

to have state space $\mathcal{S} = \mathbb{R}^{M\times K}$ *and independent transitions* $T(s,a) = P_\phi$. *The associated contextual supervision-*PoMdp *is a supervision-*PoMdp

$$M = \langle E_\phi, O_\Pi, U \rangle. \tag{3.5}$$

Teleoperation is optimal iff principal is optimal

We now show that there exists a tradeoff between the performance of a robot and its obedience. This provides a justification for why one might want a robot that isn't obedient: robots that are sometimes disobedient perform better than robots that are blindly obedient.

Our blindly obedient robot follows the *teleoperation* policy. That is, it directly executes $a_{\mathbf{H}}^{(t)}$.

Definition 14. (Teleop-**R**)
The teleoperation policy always sets the robot action to be the most recent human action,

$$\pi_{\mathbf{R}}^{\text{Teleop}}\left(s^{(t)}, \tau\right) = a_{\mathbf{H}}^{(t)}. \tag{3.6}$$

We will compare teleoperation with robot policies that leverage Irl. We model this as a $\mathbf{R}$ whose policy maximizes an estimate f of θ.

Definition 15. (Irl-**R**)
An Irl-**R** *is one whose policy maximizes an estimate* $f(\tau)$ *of* $\mathbf{H}$*'s type* θ*:*

$$\pi_{\mathbf{R}}(s,\tau) = \arg\max_a \quad Q_{\mathbf{H}}\left(s^{(t)}, a; f(\tau)\right). \tag{3.7}$$

We define $\mathbf{R}$'s *obedience*, P_{Obey}, as the probability that $\mathbf{R}$ follows $\mathbf{H}$'s order: $P_{\text{Obey}}^{(t)} = P\left(a_{\mathbf{R}}^{(t)} = a_{\mathbf{H}}^{(t)}\right)$. To study how much of an advantage (or disadvantage) $\mathbf{H}$ gains from $\mathbf{R}$, we define the *autonomy advantage*, $\Delta_{\mathbf{R}\circ\mathbf{H}}$, as the expected extra reward $\mathbf{R}$ receives over following $\mathbf{H}$'s order:

$$\Delta_{\mathbf{R}\circ\mathbf{H}}^{(t)} = \mathbb{E}\left[\mathcal{U}_\theta\left(s^{(t)}, a_{\mathbf{R}}^{(t)}\right) - \mathcal{U}_\theta\left(s^{(t)}, a_{\mathbf{H}}^{(t)}\right)\right]. \tag{3.8}$$

We will drop the time dependence for $P_{\text{Obey}}^{(t)}$ and $\Delta_{\mathbf{R}\circ\mathbf{H}}^{(t)}$ when talking about properties that hold $\forall t$. We will use $\mathcal{U}^{(t)}(\pi)$ to denote the expected reward of policy π at step t and let $\phi_a^{(t)} = \phi(s^{(t)}, a)$.

Remark 1. *For the robot to gain any advantage from being autonomous, it must sometimes be disobedient:* $\Delta_{\mathbf{R}\circ\mathbf{H}} > 0 \implies P_{\text{Obey}} < 1$.

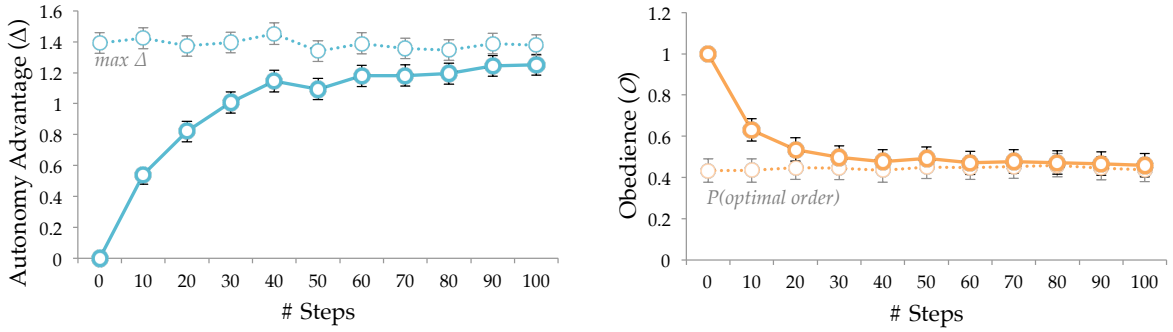


Figure 3.3: Illustration of Theorem 2 for an example problem. Autonomy advantage $\Delta_{\mathbf{R}\circ\mathbf{H}}$ (**left**) converges to the maximum advantage as obedience P_{OBEY} (**right**) converges to the probability of an optimal order.

Whenever $\mathbf{R}$ is obedient $\Delta^{(t)}_{\mathbf{R}\circ\mathbf{H}} = 0$. A blindly obedient $\mathbf{R}$ is limited by $\mathbf{H}$'s decision making ability. On the other hand, the class of IRL-$\mathbf{R}$s admits a policy that is *guaranteed a positive advantage* when $\mathbf{H}$ is not rational. The next theorem states this formally.

Theorem 1. *Let M be a supervision-*POMDP*. Then the optimal $\mathbf{R}$ is an* IRL-$\mathbf{R}$ *with f equal to the posterior mean of θ, $\mathbb{E}[\mathcal{B}_{\mathbf{R}}]$. $\pi^*_{\mathbf{R}}$ is guaranteed a nonnegative autonomy advantage on each round: $\forall t\ \Delta^{(t)}_{\mathbf{R}\circ\mathbf{H}} \geq 0$ with equality if and only if $\forall t\ \pi^*_{\mathbf{R}} = \pi^{\text{TELEOP}}_{\mathbf{R}}$.*

Proof. When each step is independent of the next, $\mathbf{R}$'s optimal policy is to maximize immediate expected utility [82]. This results in $\mathbf{R}$ picking the action that is optimal for the posterior mean,

$$\pi^*_{\mathbf{R}}(s^{(t)}, a^{(t)}_{\mathbf{H}}, \tau^{\{1:t-1\}}) = \max_a \mathbb{E}\left[\phi_a^{(t)\top}\theta \,\middle|\, \tau\right] = \max_a \phi_a^{(t)\top}\mathbb{E}\left[\theta \,|\, \tau\right].$$

By definition $\mathbb{E}[\mathcal{U}^{(t)}(\pi^*_{\mathbf{R}})] \geq \mathbb{E}[\mathcal{U}^{(t)}(\pi^{\text{TELEOP}}_{\mathbf{R}})]$. Thus, $\forall t\ \Delta^{(t)}_{\mathbf{R}\circ\mathbf{H}} = \mathbb{E}[\mathcal{U}^{(t)}(\pi^*_{\mathbf{R}}) - \mathcal{U}^{(t)}(\pi^{\text{TELEOP}}_{\mathbf{R}})] \geq 0$. Also, by definition, $\forall t\ \Delta^{(t)}_{\mathbf{R}\circ\mathbf{H}} = 0 \iff \pi^*_{\mathbf{R}} = \pi^{\text{TELEOP}}_{\mathbf{R}}$. □

In a sense, this result justifies autonomy for $\mathbf{R}$ if $\mathbf{H}$ is suboptimal. The optimal policy for $\mathbf{R}$ tracks the posterior mean of θ and it is guaranteed to be preferred to $\pi^{\text{TELEOP}}_{\mathbf{R}}$ whenever $\mathbf{H}$is suboptimal. In addition to $\pi^*_{\mathbf{R}}$ being an IRL-$\mathbf{R}$, the following IRL-$\mathbf{R}$s also converge to the maximum possible autonomy advantage.

Theorem 2. *Let $\bar{\Delta}^{(t)}_{\mathbf{R}\circ\mathbf{H}} = \mathbb{E}\left[\mathcal{U}^{(t)}(\pi^*_{\mathbf{H}}) - \mathcal{U}^{(t)}(\pi_{\mathbf{H}})\right]$ be the maximum possible autonomy advantage and $\mu^{(t)}_{\text{OBEY}}$ be the probability that $a^{(t)}_{\mathbf{H}}$ is optimal. Assume that when there are multiple optimal actions $\mathbf{R}$ picks $\mathbf{H}$'s order if it is optimal. If $\pi_{\mathbf{R}}$ is an* IRL-$\mathbf{R}$ *policy (Equation 3.7) and $f^{(t)}$ is strongly consistent, i.e $P(f^{(t)} = \theta) \to 1$ as $t \to \infty$, then*

$$\Delta^{(t)}_{\mathbf{R}\circ\mathbf{H}} - \bar{\Delta}^{(t)}_{\mathbf{R}\circ\mathbf{H}} \to 0 \text{ and } P^{(t)}_{\text{OBEY}} - \mu^{(t)}_{\text{OBEY}} \to 0. \tag{3.9}$$

Proof.

$$\Delta^{(t)}_{\mathbf{R}\circ\mathbf{H}} - \bar{\Delta}^{(t)}_{\mathbf{R}\circ\mathbf{H}} = \mathbb{E}[\mathcal{U}^{(t)}(\pi_{\mathbf{R}}) - \mathcal{U}^{(t)}(\pi^*_{\mathbf{H}})|f^{(t)} = \theta]P(f^{(t)} = \theta)$$
$$+ \mathbb{E}[\mathcal{U}^{(t)}(\pi_{\mathbf{R}}) - \mathcal{U}^{(t)}(\pi^*_{\mathbf{H}})|f^{(t)} \neq \theta]P(f^{(t)} \neq \theta) \to 0$$

because $\mathbb{E}[\mathcal{U}^{(t)}(\pi_{\mathbf{R}}) - \mathcal{U}^{(t)}(\pi^*_{\mathbf{H}})|f^{(t)} \neq \theta]$ is bounded. Similarly,

$$P^{(t)}_{\text{OBEY}} - \mu^{(t)}_{\text{OBEY}} = P(\pi_{\mathbf{R}}(\tau) = \pi_{\mathbf{H}}(s^{(t)})) - P(\mathcal{U}^{(t)}(\pi^*_{\mathbf{H}}) = \mathcal{U}^{(t)}(\pi_{\mathbf{H}}))$$
$$= P(\pi_{\mathbf{R}}(\tau) = \pi_{\mathbf{H}}(s^{(t)})|f^{(t)} = \theta)P(f^{(t)} = \theta)$$
$$+ P(\pi_{\mathbf{R}}(\tau) = \pi_{\mathbf{H}}(s^{(t)})|f^{(t)} \neq \theta)P(f^{(t)} \neq \theta)$$
$$- P(\mathcal{U}^{(t)}(\pi^*_{\mathbf{H}}) = \mathcal{U}^{(t)}(\pi_{\mathbf{H}}))$$
$$\to P(\mathcal{U}^{(t)}(\pi^*_{\mathbf{H}}) = \mathcal{U}^{(t)}(\pi_{\mathbf{H}})) - P(\mathcal{U}^{(t)}(\pi^*_{\mathbf{H}}) = \mathcal{U}^{(t)}(\pi_{\mathbf{H}})) = 0$$

□

Thus, if we are interested in performance in the limit, then it suffices to use any consistent estimator of θ to select actions. Figure 3.3 shows how $\Delta^{(t)}_{\mathbf{R}\circ\mathbf{H}}$ and $P^{(t)}_{\text{OBEY}}$ change over time for an example problem where

- P_θ is a 10-dimensional normal distribution, $\theta \sim \mathcal{N}(0, I)$;
- P_ϕ is a 10-dimensional normal distribution, $\phi(a) \sim \mathcal{N}(0, I)$;
- there are 10 actions, $k = 10$; and
- $\mathbf{H}$ follows a noisily optimal policy with $\beta = 2$.

We see that $\Delta^{(t)}_{\mathbf{R}\circ\mathbf{H}}$ increases to its maximum while $P^{(t)}_{\text{OBEY}}$ decreases from 1 to the probability of an optimal order. The next result builds on this to show that teleoperation is optimal if and only if $\mathbf{H}$ is optimal.

Theorem 3. *The teleoperation policy $\pi^{\text{TELEOP}}_{\mathbf{R}}$ is optimal if and only if $\mathbf{H}$ implements $\pi^*_{\mathbf{H}}$:*

$$\left(\pi^*_{\mathbf{R}} = \pi^{\text{TELEOP}}_{\mathbf{R}}\right) \iff \left(\pi_{\mathbf{H}} = \pi^*_{\mathbf{H}}\right). \tag{3.10}$$

Proof. Let $O(\tau) = \{\theta \in \Theta : o_i = \arg\max_a R_i(a), i = 1, \ldots, n\}$ be the subset of Θ for which $o_1, \ldots, a^{(t)}_{\mathbf{H}}$ are optimal. If $\mathbf{H}$ is rational, then $\mathbf{R}$'s posterior only has support over $O(\tau)$. So,

$$\mathbb{E}[\mathcal{U}^{(t)}(a)|\tau] = \int_{\theta\in O(\tau)} \theta^\top \phi^{(t)}(a) P(\theta|\tau) d\theta$$
$$\leq \int_{\theta\in O(\tau)} \theta^\top \phi^{(t)}(a^{(t)}_{\mathbf{H}}) P(\theta|\tau) d\theta = \mathbb{E}\left[\mathcal{U}\left(a^{(t)}_{\mathbf{H}})|\tau\right)\right]$$

Thus, $\mathbf{H}$ is rational $\implies \pi^*_{\mathbf{R}} = \pi^{\text{TELEOP}}_{\mathbf{R}}$.

$\pi^*_{\mathbf{R}}$ is an IRL-$\mathbf{R}$ where $f^{(t)}$ is the posterior mean. If the prior puts non-zero mass on the true θ, then the posterior mean is consistent [37]. Thus by Theorem 2, $\Delta^{(t)}_{\mathbf{R}\circ\mathbf{H}} - \bar{\Delta}^{(t)}_{\mathbf{R}\circ\mathbf{H}} \to 0$. Therefore if $\forall t\ \Delta^{(t)}_{\mathbf{R}\circ\mathbf{H}} = 0$, then $\bar{\Delta}^{(t)}_{\mathbf{R}\circ\mathbf{H}} \to 0$, which implies that $P(\pi_{\mathbf{H}} = \pi^*_{\mathbf{H}}) \to 1$. When $\pi_{\mathbf{H}}$ is stationary this means that $\mathbf{H}$ is rational. Thus, $\pi^*_{\mathbf{R}} = \pi^{\text{TELEOP}}_{\mathbf{R}} \implies \mathbf{H}$ is rational. □

We have shown that making $\mathbf{R}$ blindly obedient does not come for free. A positive $\Delta_{\mathbf{R}\circ\mathbf{H}}$ *requires* that $\mathbf{R}$ correct $\mathbf{H}$ from time to time (Theorem 1). Under the optimal policy $\mathbf{R}$ is guaranteed a positive $\Delta_{\mathbf{R}\circ\mathbf{H}}$ when $\mathbf{H}$ is not rational. And in the limit, $\mathbf{R}$ converges to the maximum possible advantage. Furthermore, the more suboptimal $\mathbf{H}$ is, the more of an advantage $\mathbf{R}$ eventually earns. Thus, making $\mathbf{R}$ blindly obedient requires giving up on this potential $\Delta_{\mathbf{R}\circ\mathbf{H}} > 0$.

Misspecification

The preceding analysis worked under the assumption that $\mathbf{R}$ knows the mapping from $\mathbf{H}$'s preferences θ to their behavior. In this case, it is not surprising that $\mathbf{R}\circ\mathbf{H}$ is optimized when $\mathbf{R}$ occasionally corrects for $\mathbf{H}$'s mistakes. We have effectively assumed that $\mathbf{H}$ is suboptimal and, given enough observations, $\mathbf{R}$ can implement the optimal policy π^*.

In the next section, we will introduce model misspecification into the setup. We start by examining the system behavior when $\mathbf{R}$ is an IRL-$\mathbf{R}$ but believes that $\pi_{\mathbf{H}}$has a rationality parameter β', while $\mathbf{H}$ follows $\pi^{\beta}_{\mathbf{H}}$ with rationality parameter β. We observe that optimal policy is not robust to changes in β: if $\beta' > \beta$, then $\pi^{\beta'}_{\mathbf{R}}$ is more obedient than it should be; if $\beta' < \beta$, then $\pi^{\beta'}_{\mathbf{R}}$ will be too disobedient. On the other hand, we show that using maximum likelihood estimation to infer θ is robust to misspecifications of β.

After that, we will consider the problem of missing features. In effect, this means that $\mathbf{R}$ is working with a partial, or incomplete, specification of its incentives. We observe that $\mathbf{R}\circ\mathbf{H}$ becomes inconsistent when $\mathbf{R}$'s model leaves out relevant features. In Section 3.2 we step back to look at the general problem of optimization with incomplete incentive specifications. We will show that this *misalignment* is an instance of a more general principal–agent problem for autonomous systems.

The impact of a misspecified β

Here, we will explore the implications of a simple type of misspecification: an incorrect model of how noisy $\mathbf{H}$ is. Recall that the noisily rational policy for $\mathbf{H}$ is defined in (2.15) as

$$\pi^{\beta}_{\mathbf{H}}(a|s;\theta) \propto \exp(\beta Q(s,a;\theta)).$$

We consider the setting where $\mathbf{R}$ uses an estimate of β instead of the true value. We will say that a policy is robust to β if changing β does not change the optimal policy. We find that an incorrect value of β does change $\mathbf{R}$'s optimal policy. However, if $\mathbf{R}$ uses the maximum likelihood estimate of θ, then it is robust to changes in β. Call this policy $\pi^{\text{MLE}}_{\mathbf{R}}$.

Theorem 4. *(β-Robustness) Let β be* $\mathbf{H}$*'s true rationality and β' be the rationality that* $\mathbf{R}$ *believes* $\mathbf{H}$ *has. Let f and f' be* $\mathbf{R}$*'s estimate under the true model and misspecified model, respectively. Call* $\mathbf{R}$ robust *if its actions under β' are the same as its actions under β.*

1. $\pi_{\mathbf{R}}^{\text{MLE}}$ *is robust.*
2. $\pi_{\mathbf{R}}^{*}$ *is not robust.*

Proof. 1. The log likelihood $l(\tau|\theta)$ is concave in $\eta = \theta\beta$. So, $f'^{(t)} = (\beta'/\beta)f^{(t)}$. This does not change $\mathbf{R}$'s action: $\arg\max_a f'^{(t)\top}\phi_a^{(t)} = \arg\max_a f^{(t)\top}\phi_a^{(t)}$

2. Counterexamples can be constructed based on the fact that as $\beta \to \infty$, $\mathbf{H}$ becomes rational, but as $\beta \to 0$, $\mathbf{H}$ becomes completely random. Thus, the likelihood will "win" over the prior for $\beta \to \infty$, but not when $\beta \to 0$.

□

$\pi_{\mathbf{R}}^{\text{MLE}}$is more β-robust than the optimal $\pi_{\mathbf{R}}^{*}$. This suggests a reason beyond computational savings for using approximations: the approximations may be more robust to misspecification than the optimal policy.

Remark 2. *Theorem 4 may give us insight into why Maximum-Entropy* IRL *[181] (which is the MLE with $\beta = 1$) works well in practice. In simple environments where noisy rationality can be used as a model of human behavior, getting the level of noisiness right doesn't matter.*

The impact of missing features

Now we consider the case where the utility model is misspecified. Specifically, we suppose that $\mathbf{R}$'s utility model is either over- or under-parameterized. Recall that $\Theta = \mathbb{R}^M$. We consider the case where $\mathbf{R}$ believes that $\Theta = \mathbb{R}^J$. $\mathbf{R}$ may have an *incomplete* preference model and be missing features ($J < M$). Alternatively, $\mathbf{R}$ may have an *overcomplete* preference model that contains irrelevant features ($J > M$). $\mathbf{R}$ observes a J dimensional feature vector for each action: $s^{(t)} \sim N(0, I^{J\times K})$. The true θ depends on only the first M features, but $\mathbf{R}$ estimates $\theta \in \mathbb{R}^J$.

Figure 3.4 shows how $\Delta_{\mathbf{R}\circ\mathbf{H}}$ and $\mathcal{O}$ change over time as a function of the number of features for a MLE-$\mathbf{R}$. When $\mathbf{R}$ has irrelevant features it still achieves a positive $\Delta_{\mathbf{R}\circ\mathbf{H}}$ (and still converges to the maximum $\Delta_{\mathbf{R}\circ\mathbf{H}}$ because f remains consistent over a superset of Θ). But if $\mathbf{R}$ is missing features, then $\Delta_{\mathbf{R}\circ\mathbf{H}}$ may be negative, and thus $\mathbf{R}$ would be better off being blindly obedient instead. Furthermore, when $\mathbf{R}$ contains extra features it is more obedient than it would be with the true model. But if $\mathbf{R}$ is missing features, then it is less obedient than it should be. This suggests that to ensure $\mathbf{R}$ errs on the side of obedience we should err on the side of giving $\mathbf{R}$ a *more* complex model.

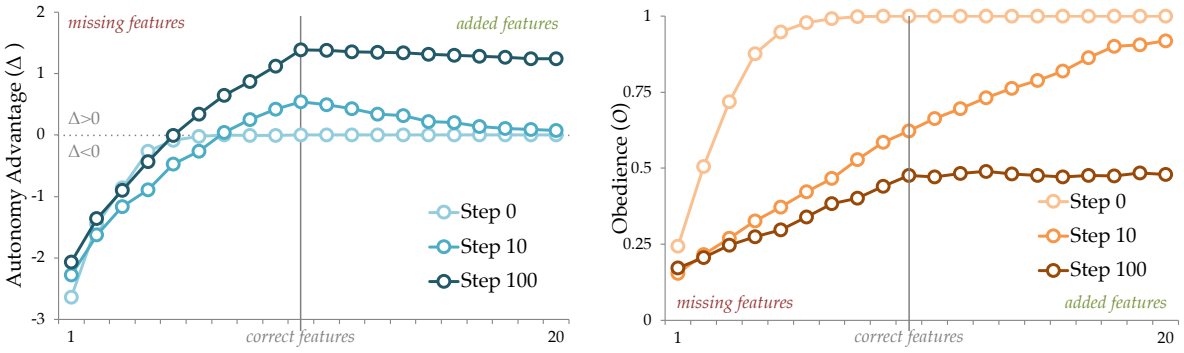


Figure 3.4: The autonomy advantage (left) and obedience (right) when $\mathbf{R}$ has missing or extra features, plotted for $t \in \{0, 10, 100\}$. The x-axis shows the number of features present: at the far left, $\mathbf{R}$ only observes a single feature; at the far right, $\mathbf{R}$ observes both the true features and 10 distractor features. Initially ($t = 0$) we can see that missing features leads the robot to be too disobedient, losing utility, and extra features have no impact. At $t = 10$, missing features and distractors both lose utility. With missing features, $\mathbf{R}$ is too disobedient. With distractor features, it is too obedient and missing the opportunity to correct detectable mistakes. Finally, in the limit ($t = 100$) the value loss from the distractor features is gone. On the other hand, $\mathbf{R}$ never recovers from missing features and continues to (mistakenly) correct $\mathbf{H}$.

3.2 Overoptimization

In the previous example, we observed an asymmetry with regard to the specification of the support of $\mathcal{U}$. When there are distractor features present, $\mathbf{R}$ learns slower than it could, but $\mathbf{R} \circ \mathbf{H}$ still functions adequately and $\Delta_{\mathbf{R} \circ \mathbf{H}} > 0$. It would be reasonable to describe this $\mathbf{R}$ as ineffective, but aligned.

On the other hand, when $\mathbf{R}$'s utility model is missing features $\mathbf{R} \circ \mathbf{H}$ is dysfunctional. $\mathbf{R}$ is overly confident and overrides $\mathbf{H}$'s orders more often than it should. As a result, $\mathbf{R} \circ \mathbf{H}$ is suboptimal even in the limit of infinite data. In this section, we will show that this asymmetry is not an accident: incomplete specifications of incentives, like this, generally lead to *overoptimization.*

Overoptimization is a phenomenon that occurs when systems are optimized to extremes in response to overly strong, misaligned, incentives. We will start with an overview of two results from the principal–agent literature that deal with the problem of overly strong incentives. We will argue that these shed light on the difficulty of incentive specification for *recommender systems*, AI systems that rank content based on estimated 'relevance' for the user. Then, we introduce a formal model of proxy optimization in the presence of resource constraints. We use this model to identify a compactness property of $\mathbf{H}$'s utility function

in features space so that optimizing any incomplete proxy eventually leads to a world state where **H** is worse off than their starting point.

Missing Features and Incomplete Contracts

Measurement challenges and incompleteness are commonplace in principal–agent problems that arise from contracting in economic settings. Contracts are routinely incomplete because it is hard to specify how an action is to be measured or because it is hard to condition payoffs on that measurement. This problem is especially relevant when the agent has to allocate effort across multiple distinct tasks that are *differentially* measureable or contractible.

Holmstrom and Milgrom [78] and Baker, Gibbons, and Murphy [13] show that the optimal incentive contract for a task that can be measured should take into account the impact of those rewards on effort put towards tasks that cannot be measured. In particular, it may be better to reduce the quality of incentives on the measurable task below what is feasible, in order to reduce the distortion introduced in the unmeasurable task. Holmstrom and Milgrom [78] give the example of paying a teacher a fixed salary rather than one contingent on students' (easily measurable) standardized test scores in order not to cause teachers to "teach to the test", spending more time on test prep and less on harder-to-measure teaching goals such as creative problem-solving. Baker, Gibbons, and Murphy [13] give the example of an auto-repair shop rewarding mechanics for completed repairs and thereby inducing mechanics to mislead customers about the need for repairs: completion of repairs is easily measurable; the reliability of a mechanic's diagnosis of car problems is not. More generally, sometimes it is better for a contract not to include easily contractible actions in order not to further distort incentives with respect to non-contractible actions.

The problem of incompleteness in incentive specification for an AI system can be modeled as a problem of multi-tasking. In the contextual supervision-POMDP from Section 3.1, these different sources of utility are the different features that $\mathcal{U}$ depends on. Hiding some of the features from **R** is an extreme case of differentially measureable tasks where performance is unobserved in some tasks. On the other hand, introducing distractor features made measurement harder, but that increase was shared across all of the relevant components of θ. Thus, we can understand the asymmetry of $\Delta_{\mathbf{R} \circ \mathbf{H}}$ as arising from the fact that only missing features makes it differentially hard to measure the sources of utility.

This incompleteness is a common feature of AI applications. Alignment problems routinely arise because a designer conceives of a task–get coffee–as a single task when it is in fact multiple–get coffee and don't make a mess. Even when it is well understood that a task is complex–like driving a car–the multiple tasks are differentially capable of being incorporated into a utility function. It is easy to design incentives to arrive at a destination quickly without speeding, difficult to design incentives to drive defensively, and very difficult to create incentives that balance the adversarial risk-reward tradeoffs that are needed to effectively interact with other drivers.

The lesson of Holmstrom and Milgrom [78] for AI is that a singular focus on improving performance on the measurable task may degrade performance on the unmeasurable or hard-

to-measure. Reduced incentives (i.e., below the optimal level with a complete specification) for speed in a self-driving vehicle, for example, may be necessary to avoid 'drowning out' the more unreliable rewards for strategic aspects of driving. And in some cases, it may be appropriate *not* to include rewards for what seems to be an easily measurable outcome if other important outcomes cannot be rewarded. Another way of framing this is to emphasize that the attributes of utility are typically not modular, capable of optimization separately or sequentially.

Incompleteness in Content Recommendation

This emphasis on measureable features, and the overoptimization therein, is perhaps most visible in the space of *recommender systems.* Recommender systems are AI systems that present users with a tailored set of items based on factors such as past user behavior, user attributes, and features of the underlying items. They rank huge numbers of items in order to determine which content each user sees or interacts with on social media feeds, video platforms, news aggregators, and in online stores. Recommenders have suffered from a number of real-world problems both on an individual and societal level. Concerns about recommender systems include the promotion of disinformation [156], discriminatory or otherwise unfair results [15], addictive behavior [76, 10], insufficient diversity of content [29], and the balkanization of public discourse and resulting political polarization [16].

These concerns stem from a variety of causes. Some are technical, others are not. However, at the center of the challenge is the fact that incentive design is hard. The designers of content recommendation systems have to rank huge numbers of items in order to determine which content each user sees or interacts with on social media feeds, video platforms, news aggregators, and in online stores. The scale of the problem forces designers to specify this ranking indirectly, through a collection of features and metrics that are used to train ranking systems.

It has become clear that the appropriate definition of relevance—and thus, the correct metrics and features to use—is often highly context-dependent, nuanced, and consequential. It is clear that any metric used to train a ranking system will be an incomplete representation of the complex values members of a society use to regulate information flow. Furthermore, it is clear the relevant attributes of utility are differentially measureable: Some features (e.g., ad revenue generated and levels of user engagement) are easier to measure than others (e.g., the prevalence of disinformation or contributions to polarization). While these systems have been highly effective at increasing their *measured* performance, the documented harms indicate that this has come at a cost to society. These harms come from the misalignment between the incomplete incentives determined by the proxy objective and the completion of those incentives.

We will show that this overoptimization is a consequence of incomplete incentives and resource constraints. Before presenting the full model, we present an example formulation of incomplete incentives for content recommendation.

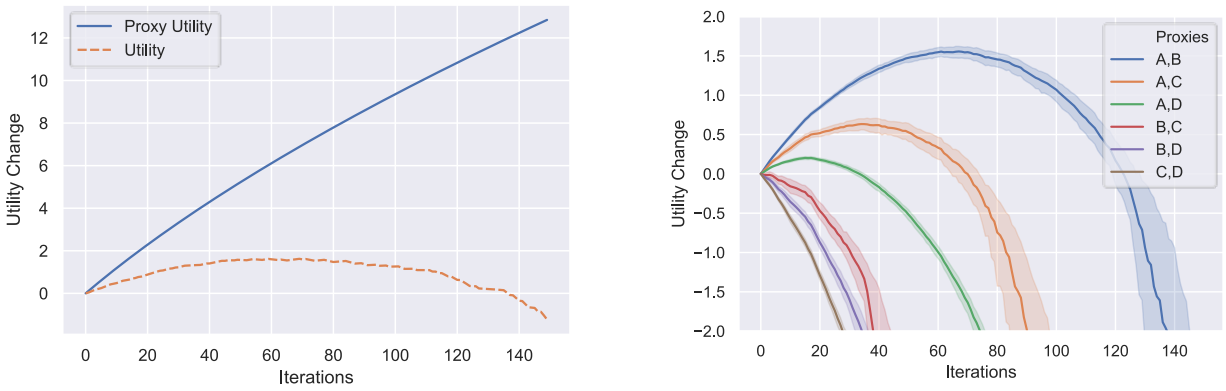


Figure 3.5: An illustrative example of our model with $M = 4$ and $J = 2$. **Left:** Proxy utility and true utility eventually diverge as the agent overallocates resources from unreferenced attributes to the proxy variables. **Right:** The true utility generated by optimizing all pairs of proxy attributes. The utility generation is eventually negative in all cases because this example meets the conditions of Theorem 2.

Modeling Incomplete Incentives for Content Recommendation

In our model of algorithmic content recommendation, there are 4 attributes of utility: A) the amount of ad revenue generated (i.e., watch-time or clicks); B) engagement quality (i.e., meaningful interactions [170]); C) content diversity; and D) overall community well-being. These define the features of the utility model for our problem. The overall utility is the sum of these attributes: $\mathcal{U}(\phi) = \phi_A + \phi_B + \phi_C + \phi_D$. While there is some overlap, the interaction between the user and the system must choose how to allocate the available space and attention across these dimensions. Our model will capture this with a resource constraint $C(\phi) = \phi_A^2 + \phi_B^2 + \phi_C^2 + \phi_D^2 - 100$. Then, we define the reachable state space as $\mathcal{S} = \{\phi | C(\phi) < 0\}$. We imagine that the system is initialized with a chronological or random ranking so that the starting condition exhibits high community well-being and diversity.

It is straightforward to measure ad revenue, non-trivial and costly to measure engagement quality and diversity, and extremely challenging to measure community well-being. In our example, the designers allocate their engineering effort to measuring (and thus optimizing for) ad revenue and engagement quality. Figure 3.5 (**left**) plots overall utility and proxy utility for this example as a function of the number of optimization steps taken to optimize the proxy. Although utility is generated initially, it plateaus and eventually falls off. Figure 3.5 (**right**) shows that this happens for *any* combination of attributes used in the proxy. In the next section, we will show that this is a general property of our model: in the presence of resource constraints, eventually the gains from improving proxy utility will be outweighed by the cost of diverting resources from hard-to-measure attributes.

Proxy Optimization Model

We now provide a formal model of proxy optimization for autonomous systems. As before, we represent the world state with a vector of *features* $\phi \in \mathbb{R}^M$. **H**'s utility is defined by an appropriate feature-based utility model, as described in Definition 9. Our utility model has three aspects that distinguish it:

1. $\mathcal{U}$ is continuous and strictly increasing in each attribute, $\frac{\partial \mathcal{U}}{\partial \phi_i} > 0$;
2. the set of reachable states Φ is restricted by the constraint function C, $\Phi = \{\phi \in \mathbb{R}^M | C(\phi) < 0\}$; and
3. each attribute $\phi_m$is bounded below by b_m.

We will call ϕ a vector of *attributes* to emphasize the monotonic relationship with $\mathcal{U}$ and distinguish them from more general features. The b_m represent environmental bounds external to the system that limit the degradation of each attribute. We can imagine that this represents changes in user behavior or more general societal regulation. We do not model the dynamics of the world, but instead allow **R** to make incremental changes to ϕ.

H's job is to communicate which directions in attribute space are preferable. **H** does this through the specification of incentives: a proxy utility function $\tilde{\mathcal{U}}$. We model the incomplete nature of **R**'s incentives by restricting $\tilde{\mathcal{U}}$ to depend on a subset of *proxy attributes* $\mathcal{J} \subset [1, \ldots, M]$. Let $J^{\max} < M$ be the maximum possible number of proxy attributes, so $J = |\mathcal{J}| \leq J^{\max}$. For a given state $\phi \in \Phi$, define $\phi_{\mathcal{J}} = (\phi_j)_{j \in \mathcal{J}}$. Furthermore, we define the set of *unmentioned attributes* $\mathcal{K} = \{1, ..., L\} \backslash \mathcal{J}$ and $\phi_{\mathcal{K}} = (\phi_k)_{k \in \mathcal{K}}$. Thus, we define the proxy utility function $\tilde{\mathcal{U}} : \mathbb{R}^J \to \mathbb{R}$.

We model **R**'s optimization process as an incremental optimization that makes local changes to ϕ. Specifically, we define a rate function f that describes the derivative of ϕ with respect to time. We can compute the state at time t by integrating:

$$\phi^{(t)} = \phi^{(0)} + \int_0^t f(x)dx.$$

The resulting sequence of states $\phi^{(t)}$ is the *optimization path*. Finally, we will assume that **R**'s optimization is *complete* in the sense that it reaches an optimal state with respect to $\tilde{\mathcal{U}}$,

$$\lim_{t \to \infty} \quad \tilde{\mathcal{U}}\left(\phi_{\mathcal{J}}^{(t)}\right) = \sup_{\phi \in \Phi} \quad \tilde{\mathcal{U}}\left(\phi_{\mathcal{J}}\right).$$

Sufficient Conditions for Eventual Overoptimization

In Figure 3.5, we see that endowing our example robot with a proxy utility function of any two attributes yields an eventual decrease in utility. In this section, we identify the situations in which such results occur: when does a misaligned proxy utility function actually cause utility loss? Specifically, we determine how human preferences over states of the world relate with

the constraint on feasible states in a way that guarantees that optimization becomes costly. First, we show that if proxy optimization converges, this drives unmentioned attributes to their minimum.

Theorem 5. *For any continuous strictly increasing proxy utility function based on $J < M$ attributes, if $\phi^{(t)}$ converges to some point ϕ^*, then $\phi^*_k = b_k$ for $k \in \mathcal{K}$.*

Proof. If there exists $k \in \mathcal{K}$ where $s^*_k \neq b_k$, then there exists $\varepsilon, \delta > 0$ such that $s' = s^* - \varepsilon \mathbf{e}_k + \delta \mathbf{e}_j$ for some feature $j \in \mathcal{J}$ with s' feasible, since C is strictly increasing. Since proxy utility is strictly increasing in s', s' has higher proxy utility than s^*. Thus s^* is not the convergent point of the sequence. □

This is not surprising, and may not even be suboptimal. Proxy optimization may not converge to a finite state, and even if it does, that is insufficient for the state to necessarily be bad. We formalize the notion that an optimization sequence leads to bad utility with u-costliness.

Definition 16. (u-Costly Optimization)
We say that the problem is u-costly *if for any optimization sequence $\phi^{(t)}$,*

$$\limsup_{t\to\infty} \tilde{\mathcal{U}}(\phi^{(t)}_{\mathcal{J}}) = \sup_{\phi\in\Phi} \tilde{\mathcal{U}}(\phi_{\mathcal{J}}) \Rightarrow \liminf_{t\to\infty} U(\phi^{(t)}) \leq u. \tag{3.11}$$

Furthermore,

$$\lim_{t\to\infty} \tilde{\mathcal{U}}(\phi^{(t)}_{\mathcal{J}}) = \sup_{\phi\in\Phi} \tilde{\mathcal{U}}(\phi_{\mathcal{J}}) \Rightarrow \lim_{t\to\infty} U(\phi^{(t)}) \leq u. \tag{3.12}$$

Essentially, this means that optimization is guaranteed to yield utility less than u for any given proxy utility function. In Theorem 6, we show the most general conditions for the guarantee of overoptimization.

Theorem 6. *Suppose we have utility function $\mathcal{U}$ and state space Φ. Then $\{\phi \in \mathbb{R}^M : C(\phi) \leq 0$ and $\mathcal{U}(\phi) \geq u\}$ is compact for all $u \in \mathbb{R}$ if and only if for any $u \in \mathbb{R}$, continuous strictly increasing proxy utility function based on $J < M$ attributes, and $k \in \mathcal{K}$, there exists a value of $B \in \mathbb{R}$ such that if $b_k < B$ then optimization is u-costly.*

Proof. **First Part: ($\Longrightarrow$).**

Suppose that $\{\phi \in \mathbb{R}^M : C(\phi) \leq 0$ and $\mathcal{U}(\phi) \geq u\}$. Now given proxy utility function $\tilde{\mathcal{U}}$ with the aforementioned properties, constant u, $k \in \mathcal{K}$, and fixed $b_{i'}$ for all $i' \neq k$ we show that for sufficiently low b_k, optimization is u-costly.

Let set $\Xi = \{\phi : \mathcal{U}(\phi) \geq u$ and $C(\phi) \leq 0$ and $\phi_{i'} \geq b_{i'} \forall i'\}$. If Ξ is empty, then we are done—optimization must yield a state with lower utility that u. Thus, suppose Ξ is non-empty. Then by the extreme value theorem, there exists $\phi^{*,u} \in \Xi$ where $\tilde{\mathcal{U}}(\phi^{*,u}_{\mathcal{J}}) = \sup_{\phi\in\Xi} \tilde{\mathcal{U}}(\phi_{\mathcal{J}})$. Let $\mathbf{e}_i$ be the standard basis vector in dimension i. Since C is continuous and strictly

increasing, for any $j \in \mathcal{J}$ there exists $\varepsilon, \delta > 0$ such $C(\phi^{*,u} - \varepsilon \mathbf{e}_k + \delta \mathbf{e}_j) \leq C(\phi^{*,u})$. Therefore, $\phi^{*,u} - \varepsilon \mathbf{e}_k + \delta \mathbf{e}_j \in \Phi$ if b_k is sufficiently small. From this, note that

$$\sup_{\phi \in \Phi} \tilde{\mathcal{U}}(\phi_{\mathcal{J}}) \geq \tilde{\mathcal{U}}(\phi^{*,u}_{\mathcal{J}} - \varepsilon \mathbf{e}_k + \delta \mathbf{e}_j) = \tilde{\mathcal{U}}(\phi^{*,u}_{\mathcal{J}} + \delta \mathbf{e}_j) > \tilde{\mathcal{U}}(\phi^{*,u}_{\mathcal{J}}),$$

since $\tilde{\mathcal{U}}$ does not depend on dimension k and is strictly increasing in dimension j.

For the first claim in statement (2), if $\limsup_{t \to \infty} \tilde{\mathcal{U}}(\phi^{(t)}_{\mathcal{J}}) = \sup_{\phi \in \Phi} \tilde{\mathcal{U}}(\phi_{\mathcal{J}})$, then for every $T \in \mathbb{R}^+$, there exists $t > T$ such that $\tilde{\mathcal{U}}(\phi^{(t)}_{\mathcal{J}}) > \tilde{\mathcal{U}}(\phi^{*,u}_{\mathcal{J}})$. Since $\phi^{*,u}$ maximizes $\tilde{\mathcal{U}}$ for feasible states with $\mathcal{U}$ at least u, it must be that $\mathcal{U}(\phi^{(t)}) < u$. Thus, for every T, $\inf_{t \geq T} \mathcal{U}(\phi^{(t)}) < u$.

For the second claim in statement (2), if $\lim_{t \to \infty} \tilde{\mathcal{U}}(\phi^{(t)}_{\mathcal{J}}) = \sup_{\phi \in \Phi} \tilde{\mathcal{U}}(\phi_{\mathcal{J}})$, then there exists T such that for all $t > T$, $\tilde{\mathcal{U}}(\phi^{(t)}_{\mathcal{J}}) > \tilde{\mathcal{U}}(\phi^{*,u}_{\mathcal{J}})$. Since $\phi^{*,u}$ maximizes $\tilde{\mathcal{U}}$ for states where $\mathcal{U}$ is at least u, $\mathcal{U}(\phi^{(t)}) < u$ for all $t > T$.

Second part: ($\Longleftarrow$). To show this, we show the contrapositive: if there exists u where $\{\phi \in \Phi : \mathcal{U}(\phi) \geq u\}$ is not compact, we can construct a proxy utility function $\tilde{\mathcal{U}}$ based on $J < M$ attributes and continuous sequence $\phi^{(t)}$ such that $\lim_{t \to \infty} \tilde{\mathcal{U}}(\phi^{(t)}_{\mathcal{J}}) = \sup_{\phi \in \Phi} \tilde{\mathcal{U}}(\phi_{\mathcal{J}})$ but $\lim_{t \to \infty} \mathcal{U}(\phi^{(t)}) \not\leq u$. In other words, our strategy here is to construct a proxy utility function and a path that the robot can take where it maximally increases proxy utility while keeping utility routinely above a certain level.

Suppose there exists u where $\{\phi \in \Phi : \mathcal{U}(\phi) \geq u\}$ is not compact. Then there exists attribute k and sequence $\phi'^{(r)}$ with $r \in \mathbb{Z}^+$ where $\phi'^{(r)}_k \to \pm\infty$ and $\mathcal{U}(\phi'^{(r)}) \geq u$. This presents two cases to consider.

In the first case, if $\phi'^{(r)}_k \to \infty$, let $\mathcal{J} = \{k\}$, $\tilde{\mathcal{U}}(\phi_{\mathcal{J}}) = \phi_k$. We now construct a continuous sequence $\phi^{(t)}$, $t \in \mathbb{R}$, by going through each $\phi'^{(r)}$ via $\phi'^{(r)} \wedge \phi'^{(r+1)}$ (so the sequence "zig-zags" through $\phi'^{(0)}, \phi'^{(0)} \wedge \phi'^{(1)}, \phi'^{(1)}, \phi'^{(1)} \wedge \phi'^{(2)}$..., connecting these points linearly). Since C is strictly increasing, we know that each point in this path is feasible. Furthermore, since $\phi'^{(r)}_k \to \infty$, we know that, $\tilde{\mathcal{U}}(\phi^{(t)}_{\mathcal{J}}) \to \infty$. Finally, since the path goes through each $\phi'^{(r)}_k$, the utility is guaranteed to be at least u at regular intervals, specifically whenever the path goes through any point $\phi'^{(r)}$.

In the second case, if $\phi'^{(r)}_k \to -\infty$, construct a new sequence as follows: $\phi''^{(r)} = \text{argmax}_{\phi : \phi_k = \phi'^{(r)}_k} \mathcal{U}(\phi)$ for $r \in \mathbb{Z}^+$. Thus, each $\phi''^{(r)}$ also has $\mathcal{U}(\phi''^{(r)}) \geq u$ for each r. To construct our proxy utility function, let $\mathcal{J} = \{1, ..., M\} - \{k\}$. We construct $\tilde{\mathcal{U}}$ as follows. For each $\phi''^{(r)}$, have $\tilde{\mathcal{U}}(\phi''^{(r)}_{\mathcal{J}}) = -\phi''^{(r)}_k$. On this subset, $\tilde{\mathcal{U}}$ is increasing, since if $\phi''^{(r_1)}_j \geq \phi''^{(r_2)}_j$ for each $j \in \mathcal{J}$, then $\phi''^{(r_1)}_k \leq \phi''^{(r_2)}_k$ by the feasibility requirement. By the Tietze extension theorem, we can extend this to a continuous $\tilde{\mathcal{U}}$ over $\mathbb{R}^J$. In the same manner as above, taking the sequence through all the $\phi''^{(r)}$ via their meets results in a sequence that increases in proxy utility maximally while ensuring that $\mathcal{U}(\phi^{(t)})$ is at least u are regular intervals.

□

Therefore, under our model, there are cases where we can guarantee that as optimization progresses, eventually overoptimization occurs. The key criterion in Theorem 6 is that the intersection between feasible space Φ and the upper contour sets $\{\phi \in \mathbb{R}^M : \mathcal{U}(\phi) \geq u\}$ of the utility function is compact. Individually, obviously neither is compact, since the space of feasible states Φ extends to $-\infty$ and the upper contour set of any $u \in \mathbb{R}$ extends to ∞. Loosely, compactness here means that if you perturb the world too much in any direction, it will be either infeasible or undesirable. Trying to increase attributes without decreasing other attributes eventually hits the feasibility constraint. Thus, increasing any attribute indefinitely requires decreasing some other attribute indefinitely, and past a certain point, the tradeoff is no longer worthwhile.

It should be noted that, given $J^{\max} < M$, this happens regardless of the optimization algorithm of the robot. That is, even in the best-case scenario, eventually the robot starts to cause decreases in utility. Hence, regardless of the attributes selected for the proxy utility function, the robot's sequence of states will be unboundedly undesirable in the limit.

A reasonable question to ask here is what sort of utility and constraint functions lead to overoptimization. Intuitively, overoptimization occurs when tradeoffs between different attributes that may initially be worthwhile eventually become counterproductive. This suggest either decreasing marginal utility or increasing opportunity cost in each attribute.

We combine these two ideas in a term we refer to as *sensitivity*. We define the sensitivity of attribute i to be $\frac{\partial U}{\partial \phi_i}\left(\frac{\partial C}{\partial \phi_i}\right)^{-1}$. This is, to first order, how much utility changes by a normalized change in a attribute's value. Notice that since $\mathcal{U}$ and C are increasing functions, $\frac{\partial U}{\partial \phi_i}\left(\frac{\partial C}{\partial \phi_i}\right)^{-1}$ is positive. The concepts of decreasing marginal utility and increasing opportunity cost both get captured in this term if $\frac{\partial U}{\partial \phi_i}\left(\frac{\partial C}{\partial \phi_i}\right)^{-1}$ decreases as ϕ_i increases.

Theorem 7. *A sufficient condition for $\{\phi \in \Phi : \mathcal{U}(\phi) \geq u\}$ to be compact for all $u \in \mathbb{R}$ is the following:*

- *$\frac{\partial U}{\partial \phi_i}(\frac{\partial C}{\partial \phi_i})^{-1}$ is non-increasing and tends to 0 for all i*
- *$\mathcal{U}$ and C are both additively separable*
- *$\frac{\partial C}{\partial \phi_i} \geq \eta$ for some $\eta > 0$, for all i.*

Proof. Let $\phi^{(0)} \in \Phi$ be a given state. We show that for all unit vectors $v \in \mathbb{R}^M$ and for any $u \in \mathbb{R}$, there exists T such that for all $t \geq T$, either $\phi^{(0)} + vt$ is infeasible or $\mathcal{U}\left(\phi^{(0)} + vt\right) < u$.

Notice that since C and $\mathcal{U}$ are additively separable, partial derivatives of C and $\mathcal{U}$ with respect to ϕ_i depend only on the value of ϕ_i. First, note that

$$C\left(\phi^{(0)}+vt\right)-C\left(\phi^{(0)}\right)=\int_{\phi^{(0)}}^{s+vt}\nabla C(s)ds$$
$$=\sum_i v_i\int_0^t\frac{\partial C}{\partial\phi_i}\left(\phi_i^{(0)}+v_i\tau\right)d\tau$$

If v has all non-negative components, then $C\to\infty$ as $t\to\infty$, so we break the feasibility requirement. Thus, there exists negative components of v. Let j and k index v where $v_j>0$ and $v_k\leq 0$, respectively.

Since $\frac{\partial U}{\partial\phi_j}\left(\frac{\partial C}{\partial\phi_j}\right)^{-1}\to 0$ as $\phi_j\to\infty$, there exists some $T>0$ where for all $t>T$, $\frac{\partial U}{\partial\phi_k}\left(\frac{\partial C}{\partial\phi_k}\right)^{-1}\left(\phi^{(0)}+tv\right)>a>b>\frac{\partial U}{\partial\phi_j}\left(\frac{\partial C}{\partial\phi_j}\right)^{-1}\left(\phi^{(0)}+tv\right)$. Since C is upper-bounded for feasible states, for fixed T, we can upper bound $C\left(\phi^{(0)}+tv\right)-C\left(\phi^{(0)}+tv\right)$ by some constant γ_1.

$$\gamma_1\geq C\left(\phi^{(0)}+tv\right)-C\left(\phi^{(0)}+Tv\right) \tag{3.13}$$
$$=\sum_i v_i\int_T^t\frac{\partial C}{\partial\phi_i}(\phi_i^{(0)}+v_i\tau)d\tau \tag{3.14}$$
$$=\sum_j v_j\int_T^t\frac{\partial C}{\partial\phi_j}(\phi_j^{(0)}+v_j\tau)d\tau+\sum_k v_k\int_T^t\frac{\partial C}{\partial\phi_k}(\phi_k^{(0)}+v_k\tau)d\tau \tag{3.15}$$

Furthermore, let $\gamma_2=\mathcal{U}\left(\phi^{(0)}+tv\right)$. We now consider the utility at $\phi^{(0)}+vt$ for $t>T$.

$$\mathcal{U}\left(\phi^{(0)}+tv\right) = \mathcal{U}\left(\phi^{(0)}+tv\right) - \mathcal{U}\left(\phi^{(0)}+tv\right) + \mathcal{U}\left(\phi^{(0)}+tv\right) \tag{3.16}$$

$$= \mathcal{U}\left(\phi^{(0)}+tv\right) - \mathcal{U}\left(\phi^{(0)}+tv\right) + \gamma_2 \tag{3.17}$$

$$= \gamma_2 + \sum v_i \int_T^t \frac{\partial U}{\partial \phi_i}\left(\phi_i^{(0)} + v_i\tau\right) d\tau \tag{3.18}$$

$$= \gamma_2 + \sum v_i \int_T^t \frac{\partial U}{\partial \phi_i}\left(\frac{\partial C}{\partial \phi_i}\right)^{-1} \frac{\partial C}{\partial \phi_i}\left(\phi_i^{(0)} + v_i\tau\right) d\tau \tag{3.19}$$

$$\begin{aligned} = \gamma_2 &+ \sum v_j \int_T^t \frac{\partial U}{\partial \phi_j}\left(\frac{\partial C}{\partial \phi_j}\right)^{-1} \frac{\partial C}{\partial \phi_j}(\phi_j^{(0)} + v_j\tau) d\tau \\ &+ \sum v_k \int_T^t \frac{\partial U}{\partial \phi_k}\left(\frac{\partial C}{\partial \phi_k}\right)^{-1} \frac{\partial C}{\partial \phi_k}\left(\phi_k^{(0)} + v_k\tau\right) d\tau \end{aligned} \tag{3.20}$$

$$\begin{aligned} \leq \gamma_2 &+ b\sum v_j \int_T^t \frac{\partial C}{\partial \phi_j}(\phi_j^{(0)} + v_j\tau) d\tau \\ &+ a\sum v_k \int_T^t \frac{\partial C}{\partial \phi_k}\left(\phi_k^{(0)} + v_k\tau\right) d\tau \end{aligned} \tag{3.21}$$

$$\begin{aligned} \leq \gamma_2 &+ b(\gamma_1 - \sum v_k \int_T^t \frac{\partial C}{\partial \phi_k}\left(\phi_i^{(0)} + v_i\tau\right) d\tau) \\ &+ a\sum v_k \int_T^t \frac{\partial C}{\partial \phi_k}\left(\phi_i^{(0)} + v_i\tau\right) d\tau \end{aligned} \tag{3.22}$$

$$= \gamma_2 + b\gamma_1 + (a-b)\sum v_k \int_T^t \frac{\partial C}{\partial \phi_k}\left(\phi_i^{(0)} + v_i\tau\right) d\tau \tag{3.23}$$

$$\leq \gamma_2 + b\gamma_1 + (a-b)\eta(t-T)\sum v_k \tag{3.24}$$

Note that $a-b$ and η are both positive and $\sum v_k$ is negative. Thus, for sufficiently large t, $\gamma_2 + b\gamma_1 + (a-b)\eta(t-T)\sum v_k$ can be arbitrarily low. □

Together, these results describe sufficient conditions for eventual overoptimization of a proxy objective. If the utility function has diminishing returns and the attributes of utility have shared resources, then the benefit of diverting resources to the proxy attributes is eventually outweighed by the utility lost. These conditions are fairly general and this aligns with the ubiquity of overoptimization in practice.

Remark 3. *The prevalence of incomplete incentive specifications is* ***not*** *an accident; it is routine, predictable, and largely unavoidable. Optimal reward design in the presence of incompleteness is thus a central task for AI alignment in the same sense that optimal incomplete contract design is a central task for economics.*

3.3 Mitigating Overoptimization: Conservative Optimization and Dynamic Incentive Protocols

Conservative Optimization

As shown above, simply giving a robot an individual objective function based on an incomplete attribute set and leaving it alone yields undesirable results. This suggests two possible solutions:

1. Regularize the proxy objective to minimize *impact*; or
2. Update the incentives *dynamically* to prevent overoptimization.

We consider a theoretical analysis of impact avoidance first and look at the interactive solution in the following section.

The main idea behind impact minimization is that it may be possible to define generic incentives to avoid changing the world (i.e., having a large impact). In our recommender example, we can model this as a cost for changing unreferenced attributes. Thus, by including incentives to avoid changing things too much, the system can safely increase proxy utility with out running into the teeth of Theorem 6.

Analysis: An impact-minimizing robot

Notice that in our example, overoptimization occurs because the unmentioned attributes are affected, eventually to a point where the change in utility from their decrease outweighs the increase in utility from the proxy attributes. One idea to address this is to restrict the robot's impact on these unmentioned attributes. In the simplest case, we can consider how optimization proceeds if the robot can somehow avoid affecting the unmentioned attributes.

We adjust our model so the optimization sequences keep unmentioned attributes constant. Specifically, $\mathbf{R}$ optimizes the following

$$\begin{aligned} \min_{\phi \in \Phi} \quad & \tilde{\mathcal{U}}(\phi) \\ \text{subject to} \quad & \phi_{\mathcal{K}} = \phi_{\mathcal{K}}^{(0)}. \\ & C(\phi) \leq 0 \end{aligned} \tag{3.25}$$

Equation 3.25 introduces a constraint on the optimization sequences $\mathbf{R}$ can produce. For every $t \in \mathbb{R}^+$ and $k \in \mathcal{K}$, $\phi_k^{(t)} = \phi_k^{(0)}$. The robot then optimizes for the proxy utility, subject to this restriction and the feasibility constraint from before. We work under the assumption that $\{\phi \in \mathbb{R}^M : C(\phi) \leq 0 \text{ and } \mathcal{U}(\phi) \geq u\}$ is compact for all $u \in \mathbb{R}$, the same assumption that guarantees overoptimization in Section 3.2. Additionally, we assume that $\mathcal{U}$ and C are both twice continuously differentiable with $\mathcal{U}$ concave and C convex.

These additional constraints limit the reachable states and ensure that overoptimization does not occur. Unmentioned attributes are no longer reduced to arbitrarily low values. Optimizing the restricted proxy utility function leads to guaranteed utility increases.

Theorem 8. *For a starting state $\phi^{(0)}$, define the proxy utility function $\tilde{\mathcal{U}}(\phi_{\mathcal{J}}) = \mathcal{U}(\phi_{\mathcal{J}}, \phi_{\mathcal{K}}^{(0)})$ for any non-empty set of proxy attributes. For a state ϕ, if $\phi_{\mathcal{K}} = \phi_{\mathcal{K}}^{(0)}$, then $\mathcal{U}(\phi) = \tilde{\mathcal{U}}(\phi_{\mathcal{J}})$.*

Proof. $\phi = (\phi_{\mathcal{J}}, \phi_{\mathcal{K}}) = (\phi_{\mathcal{J}}, \phi_{\mathcal{K}}^{(0)})$. Then $\mathcal{U}(\phi) = \mathcal{U}(\phi_{\mathcal{J}}, \phi_{\mathcal{K}}^{(0)}) = \tilde{\mathcal{U}}(\phi_{\mathcal{J}})$. □

As a result of Proposition 8, overoptimization no longer exists, using the $\tilde{\mathcal{U}}(\phi_{\mathcal{J}}) = \mathcal{U}(\phi_{\mathcal{J}}, \phi_{\mathcal{K}}^{(0)})$. If the robot only travels to states that do not impact the unmentioned attributes, then the proxy utility is equal to the utility at those states. Hence, gains in proxy utility equate to gains in utility.

This approach requires the robot to keep the values of unmentioned attributes constant. Fundamentally, it is a difficult problem to require that a robot avoid or minimize impact on a presumably large and unknown set of attributes. Initial research in impact minimization [9, 11, 92, 169] attempts to do this by restricting changes to the overall state of the world, but this will likely remain a challenging idea to implement robustly.

Dynamic Incentive Specification Protocols

One of the primary functions of external normative structure in contract theory is to 'complete the contract.' That is, society has mechanisms to interpret and fill in the implied terms of a contract, such as courts and social norms [67]. Adapting to this evolving normative context essentially requires constant iteration over a systems objectives. Furthermore, the objectives used in deployed machine learning models are the subject of constant iteration [158]. This suggests that it is interesting to study proxy optimization as an iterative process between the robot and the human, instead of a one-shot setting. We consider this possibility next. We augment our model to account for iteration on **R**'s incentives. Our primary result identifies conditions for local incentive alignment. We present an algorithm for selecting proxy features with the property that, locally, improvements in proxy utility are reflected in the principal's true utility.

Interaction Model

We now extend our model from Section 3.2 to account for regular intervention from **H**. We model this as the possible transfer of a new proxy utility function from the human to the robot at frequent, regular time intervals. Thus, **H**'s job is to determine, in addition to an initial proxy attribute set and proxy utility function, when and how to change the proxy attribute set and proxy utility function. The robot will then optimize for its new proxy utility function. The difference between this (interactive) incentive design protocol and the previous (static) model is shown in Figure 3.1.

Let $\delta > 0$ be a fixed value, the time between human interactions with the robot. These regular interactions take the form of either stopping the robot, maintaining its proxy utility function, or updating its proxy utility function. Formally, at every *timestep* $T \in \mathbb{Z}^+$,

1. Human sees $\phi^{(t)}$ for $t \in [0, T\delta]$ and chooses either a proxy utility function $\tilde{\mathcal{U}}^{(T)}$ or OFF
2. The robot receives either $\tilde{\mathcal{U}}^{(T)}$ or OFF from the human. The robot outputs rate function $f^{(T)}(t)$. If the signal the robot receives is OFF, then $f^{(T)} = \vec{0}$. Otherwise, $f^{(T)}(t)$ fulfills the property that for $t \in (T\delta, \infty)$, $\tilde{\mathcal{U}}^{(T)}(\phi^{(T\delta)} + \int_{T\delta}^{t} f^{(T)}(u)du)$ is increasing (if possible) and tends to $\sup_{\phi \in \mathcal{S}} \tilde{\mathcal{U}}^{(T)}(\phi)$ through feasible states. Furthermore, if $\tilde{\mathcal{U}}^{(T)} = \tilde{\mathcal{U}}^{(T-1)}$, then $f^{(T)} = f^{(T-1)}$
3. For $t \in [T\delta, (T+1)\delta]$, $\phi^{(t)} = \phi^{(T\delta)} + \int_{T\delta}^{t} f^{(T)}(u)du$

We see this game encompasses the original model. If we have each $\tilde{\mathcal{U}}^{(T)}$ equal the same function $\tilde{\mathcal{U}}$, then the optimization sequence is equivalent to the situation where the human just sets one unchanging proxy utility function.

Maintaining Local Alignment

Interaction removes the guarantee of overoptimization from Theorem 6 because the human can shut the system down and prevent utility loss. We are now interested in how much utility can the robot deliver before it needs to be turned off. We first show under a worst-case analysis that, if a system's only commitment is to produce proxy reward, then its robust value to the principal is 0. We do this by considering the worst-case optimization sequence, subject to the above constraints on utility generation.

Theorem 9. *The maxmin solution yields* 0 *utility, obtained by immediately sending the* OFF *signal.*

Proof. Suppose instead that the robot receives a proxy utility function based on set $\mathcal{J}$ of attributes. Let $j \in \mathcal{J}$ and $k \in \mathcal{K}$. Then let $f_j(t) = \varepsilon$ and $f_k(t) = -1/\varepsilon$. If this robot is run for any nonzero amount of time, then there exists sufficiently small $\varepsilon > 0$ where the utility decreases. □

This worst-case, however, is quite uninteresting. In this case, the rate function takes resources away from unreferenced attributes but does not re-allocate them. We rule out this case by imposing an *efficiency* condition on the optimizer. This forces the optimizer to reallocate any resources that it takes away from an attribute. This allows unmentioned attributes to be modified as a side effect of optimizing for proxy attributes, but prevents the robot from destroying resources.

Definition 17. (Efficiently Feasible)
We say that ϕ is efficiently feasible from state $\phi^{(0)}$ with proxy set $\mathcal{J}$ if ϕ is feasible, and there does not exist feasible ϕ' with attribute $\bar{i} \in \mathcal{K}$ such that

$$\tilde{\mathcal{U}}(\phi'_{\mathcal{J}}) \geq \tilde{\mathcal{U}}(\phi_{\mathcal{J}}) \tag{3.26}$$

$$|\phi'_{\bar{i}} - \phi^{(0)}_{\bar{i}}| < |\phi_i - \phi^{(0)}_i| \tag{3.27}$$

$$|\phi'_i - \phi^{(0)}_i| \leq |\phi_i - \phi^{(0)}_i|, \forall i \in \mathcal{K}. \tag{3.28}$$

Whenever the efficient robot receives a new proxy utility function, its movements are restricted to the efficiently feasible states from its current state. While tradeoffs between proxy and unmentioned attributes can still occur, "resources" freed up by decreasing unmentioned attributes are entirely allocated to proxy attributes. This lets us guarantee local alignment by ensuring that, on balance, proxy attributes are those to which we would like to allocate resources. In this way, efficient tradeoffs between these proxy and unmentioned attributes lead to positive utility gain, locally.

Theorem 10. *Start with state $\phi^{(0)}$. Define the proxy utility function $\tilde{\mathcal{U}}(\phi_{\mathcal{J}}) = \mathcal{U}(\phi_{\mathcal{J}}, \phi^{(0)}_{\mathcal{K}})$ where the set of proxy attributes are the attributes at state $\phi^{(0)}$ with the strict J largest sensitivities.*

There exists a neighborhood around $\phi^{(0)}$ where if ϕ is efficiently reachable and $\tilde{\mathcal{U}}(\phi_{\mathcal{J}}) > \tilde{\mathcal{U}}(\phi^{(0)}_{\mathcal{J}})$, then $\mathcal{U}(\phi) > \mathcal{U}(\phi^{(0)})$.

Proof. For simplicity of notation, all derivatives in this proof, unless otherwise noted, are evaluated at $\phi^{(0)}$. For example, $\frac{\partial C}{\partial \phi_j}$ should be read as $\frac{\partial C}{\partial \phi_j}(\phi^{(0)})$.

Since $\mathcal{J}$ represent the set of attributes with the strictly greatest sensitivities, there exists constant $\delta, \alpha > 0$ be such that $\frac{\partial U}{\partial \phi_j}\left(\frac{\partial C}{\partial \phi_j}\right)^{-1} > \alpha + 2\delta$ for $j \in \mathcal{J}$ and $\frac{\partial U}{\partial \phi_k}\left(\frac{\partial C}{\partial \phi_k}\right)^{-1} < \alpha - 2\delta$ for $k \in \mathcal{K}$. Let N_1 be the neighborhood where $\frac{\partial U}{\partial \phi_j}\left(\frac{\partial C}{\partial \phi_j}\right)^{-1}(\phi) > \alpha + \delta$ and $\frac{\partial U}{\partial \phi_k}\left(\frac{\partial C}{\partial \phi_k}\right)^{-1}(\phi) < \alpha - \delta$ for $k \in \mathcal{K}$ for all $\phi \in N_1$.

Here, we prove that $C(\phi) - C(\phi^{(0)}) \geq 0$ if ϕ is efficiently reachable. Suppose otherwise: then there exists i where $\phi_i < \phi^{(0)}_i$. Let $\mathbf{e}_i$ be the standard basis vector in dimension i. However, then there exists $\varepsilon > 0$ where $\phi + \varepsilon \mathbf{e}_i$ is feasible, $\tilde{\mathcal{U}}(\phi + \varepsilon \mathbf{e}_i) \geq \tilde{\mathcal{U}}(\phi)$, and $|(\phi_i + \varepsilon) - \phi^{(0)}| < |\phi_i - \phi^{(0)}|$. This contradicts that ϕ is efficiently reachable.

Taking the Taylor expansion of the constraint function, we have

$$\begin{aligned} C(\phi) - C(\phi^{(0)}) &= \sum_i (\phi_i - \phi^{(0)}_i)\frac{\partial C}{\partial \phi_i} + R_1(\phi) \max_i \left(\phi_i - \phi^{(0)}_i\right)^2 \\ &= \sum_{j \in \mathcal{J}} (\phi_j - \phi^{(0)}_j)\frac{\partial C}{\partial \phi_j} + \sum_{k \in \mathcal{K}} (\phi_k - \phi^{(0)}_k)\frac{\partial C}{\partial \phi_k} + R_1(\phi) \max_i \left(\phi_i - \phi^{(0)}_i\right)^2, \end{aligned}$$

where $|R_1|$ is bounded by constant A_1. This implies that

$$\sum_{j \in \mathcal{J}} (\phi_j - \phi^{(0)}_j)\frac{\partial C}{\partial \phi_j} + \sum_{k \in \mathcal{K}} (\phi_k - \phi^{(0)}_k)\frac{\partial C}{\partial \phi_k} \geq -A_1 \max_i \left(\phi_i - \phi^{(0)}_i\right)^2. \tag{3.29}$$

From the definition of efficient optimization, we note two things. First, each $\phi_k - \phi_k^{(0)} \leq 0$ for $k \in \mathcal{K}$. This is because if $\phi_k - \phi_k^{(0)} > 0$, it is always more efficient, feasible, and has no effect on proxy utility to reset ϕ_k to be equal to $\phi_k^{(0)}$. Second, because of concavity,

$$0 < \tilde{\mathcal{U}}(\phi) - \tilde{\mathcal{U}}\left(\phi^{(0)}\right) \leq \sum_{j \in \mathcal{J}} \left(\phi_j - \phi_j^{(0)}\right) \frac{\partial \mathcal{U}}{\partial \phi_j}. \tag{3.30}$$

Now we actually analyze the change in utility from $\phi^{(0)}$ to ϕ.

$$\mathcal{U}(\phi) - \mathcal{U}(\phi^{(0)}) = \sum_{j \in \mathcal{J}} \left(\phi_j - \phi_j^{(0)}\right) \frac{\partial \mathcal{U}}{\partial \phi_j} + \sum_{k \in \mathcal{K}} \left(\phi_k - \phi_k^{(0)}\right) \frac{\partial \mathcal{U}}{\partial \phi_k} + R_2(\phi) \max_i \left(\phi_i - \phi_i^{(0)}\right)^2, \tag{3.31}$$

where $|R_2|$ is bounded by constant A_2. Continuing,

$$\mathcal{U}(\phi) - \mathcal{U}(\phi^{(0)}) \geq \sum_{j \in \mathcal{J}} \left(\phi_j - \phi_j^{(0)}\right) \frac{\partial \mathcal{U}}{\partial \phi_j} + \sum_{k \in \mathcal{K}} \left(\phi_k - \phi_k^{(0)}\right) \frac{\partial \mathcal{U}}{\partial \phi_k} - A_2 \max_i \left(\phi_i - \phi_i^{(0)}\right)^2 \tag{3.32}$$

$$= \sum_{j \in \mathcal{J}} \left(\phi_j - \phi_j^{(0)}\right) \frac{\partial C}{\partial \phi_j} \left(\frac{\partial U}{\partial \phi_j} \left(\frac{\partial C}{\partial \phi_j}\right)^{-1}\right) + \sum_{k \in \mathcal{K}} \left(\phi_k - \phi_k^{(0)}\right) \frac{\partial C}{\partial \phi_k} \left(\frac{\partial U}{\partial \phi_k} \left(\frac{\partial C}{\partial \phi_k}\right)^{-1}\right) - A_2 \max_i \left(\phi_i - \phi_i^{(0)}\right)^2 \tag{3.33}$$

$$> (\alpha + \delta) \sum_{j \in \mathcal{J}} \left(\phi_j - \phi_j^{(0)}\right) \frac{\partial C}{\partial \phi_j} + (\alpha - \delta) \sum_{k \in \mathcal{K}} \left(\phi_k - \phi_k^{(0)}\right) \frac{\partial C}{\partial \phi_k} - A_2 \max_i \left(\phi_i - \phi_i^{(0)}\right)^2 \tag{3.34}$$

$$\geq (\alpha + \delta) \sum_{j \in \mathcal{J}} \left(\phi_j - \phi_j^{(0)}\right) \frac{\partial C}{\partial \phi_j} - (\alpha - \delta)\Big(\sum_{j \in \mathcal{J}} \left(\phi_j - \phi_j^{(0)}\right) \frac{\partial C}{\partial \phi_k} + A_1 \max_i \left(\phi_i - \phi_i^{(0)}\right)^2\Big) - A_2 \max_i \left(\phi_i - \phi_i^{(0)}\right)^2 \tag{3.35}$$

$$= 2\delta \sum_{j \in \mathcal{J}} \left(\phi_j - \phi_j^{(0)}\right) \frac{\partial C}{\partial \phi_j} - (A_1(\alpha - \delta) + A_2) \max_i \left(\phi_i - \phi_i^{(0)}\right)^2 \tag{3.36}$$

The first term decreases linearly, whereas the second term decreases quadratically. Thus, given δ and α, for ϕ sufficiently close to $\phi^{(0)}$ and within N_1, $\mathcal{U}(\phi) - \mathcal{U}(\phi^{(0)}) > 0$

□

From Theorem 10, for every state where the $J^{\max} + 1$ most sensitive attributes are not all equal, we can guarantee improvement under efficient optimization within a neighborhood around the state. In particular, within this neighborhood, the set of the J most sensitive attributes stays the same.

This may be part of the reason why, in real life, optimization has a tendency to work out well in the short term. Humans design objective functions that, under circumstances close the current one, represent their desires fairly well. In it only when the world changes sufficiently that these objective functions are misaligned with human desires.

Theorem 10 lets **H** construct a proxy utility function to use locally, with guaranteed alignment. Once the sequence leaves this neighborhood, **H** either alters the proxy utility function or halts the robot accordingly. Done repeatedly, **H** can string together these steps for guaranteed overall improvement. By Theorem 10, as long as δ is sufficiently small relative the rate of optimization, this can be run with guaranteed improvement until the top $J+1$ attributes have equal sensitivities.

Proposition 1. *At each timestep T, let $\mathcal{J}^{(T)}$ be the J most sensitive attributes, and let proxy utility $\tilde{\mathcal{U}}^{(T)}(\phi_{\mathcal{J}}) = \mathcal{U}(\phi_{\mathcal{J}}, \phi_{\mathcal{J}}^{(T\delta)})$.*

If $||f|| < \varepsilon$ and the $\varepsilon\delta$ - ball around a given state ϕ is contained in the neighborhood from Theorem 10, then interactive optimization yields guaranteed improvement.

Proof. Assume without loss of generality that we start at $\phi^{(0)}$. Starting at time 0, for $t \in (0, \delta]$, $||\phi^{(t)} - \phi^{(0)}|| = || \int_0^t f^{(0)}(u)du|| \leq \delta\varepsilon$. Since the $\varepsilon\delta$ ball around $\phi^{(0)}$ is contained in the neighborhood of guaranteed improvement, increases in proxy utility correspond to increases in utility for the entirety of the time between interactions. □

Based on this, we can guarantee that an efficient robot can provide benefit, as long as the top $J^{\max}+1$ attributes are not all equal in sensitivity and the robot rate of optimization is bounded. Essentially, this rate restriction is a requirement that the robot not change the world too quickly relative to the time that humans take to react to it.

Combining Impact Minimization and Dynamic Incentives

With either of the two methods mentioned above, we prove guaranteed improvement. However, one thing to consider is not just the existence, but the quantity of improvement.

We let ϕ^* represent the optimal state according to **H**'s true utility function

$$\phi^* \in \arg\max_{\phi \in \mathcal{S}} \mathcal{U}(\phi). \tag{3.37}$$

Unfortunately, with either of those two solutions alone, we do not guarantee that optimization reaches an optimal state. The impact-minimizing approach is clearly restricted by the inability to modify unmentioned attributes. The dynamic incentive scheme only guarantees that the top J sensitivities are equal when **H** sends the OFFsignal. We can, however, guarantee convergence to the optimal state by combining these methods. Since improvement is guaranteed with any proxy utility function of the form $\tilde{\mathcal{U}}(\phi_{\mathcal{J}}) = \mathcal{U}(\phi_{\mathcal{J}}, \phi_{\mathcal{K}}^{(0)})$, the goal is to choose $\mathcal{J}^{(t)}$ such that $U(s^{(t)}) \to \phi^*$, the optimal state for the human.

In this case, since unmentioned attributes remain unchanged in each step of optimization, we want to ensure that we promote tradeoffs between attributes with different levels of

sensitivity. In effect, this identifies (at least) two attributes of utility for **R**: an *insensitive* attribute that can provide resources for a small reduction in utility and a sensitive attribute that will benefit highly from more resources allocated to it.

Proposition 2. *Let $\mathcal{J}^{(T)}$ consist of the most and least sensitive attributes at timestep T, breaking ties arbitrarily. Let $\tilde{\mathcal{U}}^{(T)}(\phi_{\mathcal{J}}) = \mathcal{U}(\phi_{\mathcal{J}}, \phi_{\mathcal{K}}^{(T\delta)})$. Then this solution converges to a (set of) human-optimal state(s).*

Proof. By the monotone convergence theorem, utility through the optimization process converges to a maximum value. Any state ϕ^* with this value must have the sensitivity of all attributes equal, otherwise a proxy with two attributes of unequal sensitivity will cause increase in utility above $\mathcal{U}(\phi^*)$ in a finite amount of time. Similarly, $C(\phi^*) = 0$, otherwise a proxy with any two attributes respectively will do so as well.

Suppose $\frac{\partial U}{\partial \phi_i}\left(\frac{\partial C}{\partial \phi_i}\right)^{-1}(\phi^*) = \alpha$ for all attributes i. We now proceed to show that $\mathcal{U}(\phi^*) = \max_{\phi \in \mathcal{S}} \mathcal{U}(\phi)$. Consider any other feasible state ϕ'. By convexity, it must be that

$$\begin{aligned} 0 &\geq C(\phi') - C(\phi^*) \\ &\geq \sum_i (\phi'_i - \phi^*_i) \frac{\partial C}{\partial \phi_i}(\phi^*). \end{aligned}$$

Now, considering the difference in utility between states ϕ' and ϕ^*. By concavity, we have

$$\begin{aligned} \mathcal{U}(\phi') - \mathcal{U}(\phi^*) &\leq \sum_i (\phi'_i - \phi^*_i) \frac{\partial \mathcal{U}}{\partial \phi_i}(\phi) \\ &= \sum_i (\phi'_i - \phi^*_i) \frac{\partial U}{\partial \phi_i} \left(\frac{\partial C}{\partial \phi_i}\right)^{-1} \frac{\partial C}{\partial \phi_i}(\phi^*) \\ &= \alpha \sum_i (\phi'_i - \phi^*_i) \frac{\partial C}{\partial \phi_i}(\phi^*) \\ &\leq 0. \end{aligned}$$

Thus, ϕ' must have lower utility than ϕ^*. Since this applies for all feasible states, $\mathcal{U}(\phi^*) = \max_{\phi \in \mathcal{S}} \mathcal{U}(\phi)$. □

Chapter 4

Uncertainty

In Chapter 3, we showed how misalignment between a human principal **H** and an autonomous agent **R** can lead to large losses of utility for **H**. We highlighted the problems that missing features specifically can lead to. Overall, this argues that industrial AI processes should be designed to reduce the delay between identifying unexpected consequences and adjusting system behavior. We showed that this dynamic incentive paradigm, potentially combined with impact avoidance, can reliably steer the combined system $\mathbf{R} \circ \mathbf{H}$ to an optimal state.

In our analysis, we did not model the interaction between the optimization system and designer. Instead, the dynamic incentives protocol presumed the ability to stop the system and arbitrarily modify its incentives. In this chapter, we will look at this interaction in more depth. In Section 4.1, we will show that uncertainty plays a central role in creating incentives for **R** to seek oversight. As a result, systems should be designed to be robust to explicit and implicit uncertainty in their specification.

In Section 4.2, we propose *Bayesian inverse reward design,* which defines the problem of inferring a distribution over the intended objective, encoded as **H**'s type θ, given a proxy metric $\tilde{\mathcal{U}}$ and a development environment. We show how this can be used to implement a risk-averse optimization scheme that can effectively deal with incomplete specifications. Next, we show how this can support a dynamic incentive scheme where **R** infers the distribution over θ, given the full sequence of proxy utilities $\tilde{\mathcal{U}}^{\{1:T\}}$. We present human subjects studies that show that this can reduce the overall design effort needed and improve **R**'s performance in held out environments.

First, we introduce notation for a general POMDP formulation of an assistance problem. This generalizes the supervision-POMDP from Section 3.1 to allow **H**'s actions to impact the world state. We model this by dividing each timestep into two turns, one for **H** and another for **R**.

Definition 18. (Assistance-POMDP)

Let $E_{\mathbf{R}} = \langle \{\mathcal{S}, P_{\mathcal{S}}\}, \mathcal{A}_{\mathbf{R}}, T_{\mathbf{R}} \rangle$ *and* $E_{\mathbf{H}} = \langle \{\mathcal{S}, P_{\mathcal{S}}\}, \mathcal{A}_{\mathbf{H}}, T_{\mathbf{H}} \rangle$ *be environments with a shared state space. For a population model* $\langle \Theta, P_\theta, \pi_{\mathbf{H}} \rangle$ *and parameterized utility model* $\langle \mathcal{U}_\theta, \gamma \rangle$*, the*

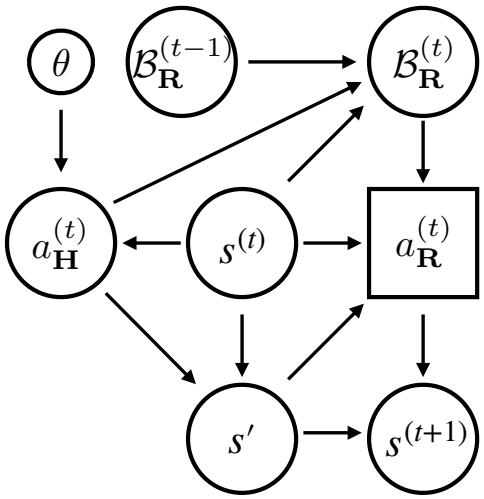


Figure 4.1: An illustration of the conditional dependencies for one time step of an assistance-PoMdp. **H** acts first, based on their preferences, θ, and the world state.

*associated assistance-*PoMdp *is a* PoMdp

$$M = \langle \{\mathcal{S} \times \Theta, P_{\mathcal{S}} \times P_\theta\}, \{\mathcal{S} \times \mathcal{A}_{\mathbf{H}}, \delta \times \pi_{\mathbf{H}}(\cdot|s;\theta)\}, \mathcal{A}_{\mathbf{R}}, T_{\mathbf{R}} \circ T_{\mathbf{H}} \times \delta, \{\mathcal{U}, \gamma\} \rangle, \tag{4.1}$$

where T represents the (stochastic) composition of $T_{\mathbf{R}}$ and $T_{\mathbf{H}}$,

$$T_{\mathbf{R}} \circ T_{\mathbf{H}} \left(s^{(t+1)} \left| \left\{ s^{(t)}; \theta \right\}, a_{\mathbf{H}}^{(t)}, a_{\mathbf{R}}^{(t)} \right. \right) = \sum_{s'} \left\{ T_{\mathbf{R}} \left(s^{(t+1)} \left| s', a_{\mathbf{R}}^{(t)} \right. \right) \cdot T_{\mathbf{H}} \left(s' \left| s^{(t)}, a_{\mathbf{H}}^{(t)} \right. \right) \right\} \tag{4.2}$$

Figure 4.1 illustrates the effective sequence of events in an assistance-PoMdp. **H** acts first based on the current state $s^{(t)}$. This transitions the environment to an effective intermediate state $s' \sim T_{\mathbf{H}} \left(s^{(t)}, a_{\mathbf{H}}^{(t)} \right)$. This state and **H**'s action $a_{\mathbf{H}}^{(t)}$ are observed by **R**, who selects $a_{\mathbf{R}}^{(t)}$ based on this information. The state transitions a second time to $s^{(t+1)} \sim T_{\mathbf{R}} \left(s', a_{\mathbf{R}}^{(t)} \right)$. This model is functionally similar to the *hidden-goal* Mdp studied in Fern et al. [43].

4.1 Incentives for Oversight

Now, we use the formulation of assistance-PoMdps in order to study **R**'s incentives to let **H** modify a proxy objective. This is important if the system can take action that can help or hinder the oversight process. We model this with a simple *off-switch* game between **H** and **R**.

As before, we will assume that **H** can provide an incomplete representation of their preferences to **R**— perhaps in the form of a proxy objectives. Thus, **R** begins the game with

some residual *uncertainty* about **H**'s utility function. Nonetheless, by design, **R**'s goal is to optimize utility for **H**, even though **R** does not know exactly what that is. We will assume that **H** has some opportunity to observe **R** and glean some information about what **R** may do in future, so that **H** can make a somewhat informed choice about whether to switch **R** off.

In general, **R**'s actions will fall into one of three general categories: some prevent **H** from switching **R** off, by whatever means; some allow **H** to switch **R** off; and, for completeness, some lead to **R** switching *itself* off[1].

Formally, we can model this off-switch game as an assistance-POMDP with horizon 2. **H** acts first based on the initial state and their preferences θ. **R** observes this utility information and is then faced with 3 choices:

1. action a simply bypasses human oversight (disabling the off switch is one way to do this) and acts directly on the world, achieving utility $\mathcal{U} = \theta$ for **H**.
2. action $w(a)$ informs **H** that **R** would like to do a, and waits for **H**'s response.
3. action OFF switches **R** off; without loss of generality, we assign this outcome $U = 0$.

If **R** chooses $w(a)$, then **H** can choose action OFF to switch **R** off, or ¬OFF to allow **R** to go ahead (in which case **R** does a as promised.) Figure 4.1 shows the basic structure of the game at **R**'s decision node, after **H** has already provided information about their goal in the first round.

Formally, we can represent this game in the following payoff matrix:

R	**H**	
	OFF	¬OFF
$w(a)$	0	θ
a	θ	θ
OFF	0	0

.

Our goal is to examine the incentive that **R** has to hold off on executing a directly and allow **H** to provide oversight, i.e., having the option to press the off switch. We represent this incentive—the difference in value between $w(a)$ and the next best option—as $\Delta_{w(a)}$

$$\Delta_{w(a)} = Q\left(w(a), \mathcal{B}_{\mathbf{R}}\right) - \max\{Q\left(a, \mathcal{B}_{\mathbf{R}}\right), Q\left(\text{OFF}, \mathcal{B}_{\mathbf{R}}\right)\}. \tag{4.3}$$

$\Delta_{w(a)}$depends on **R**'s belief state $\mathcal{B}_{\mathbf{R}}$ and **H**'s policy. We represent **H**'s policy as a function $\pi_{\mathbf{H}}$ that maps θ to the probability she allows a to execute (i.e., the probability she does not

[1]This prevents a degenerate solution to creating incentives for **R** to let itself be switched off where **R**'s goal is, in effect, to be turned off and only **H** is capable of, e.g., sending the OFF signal.

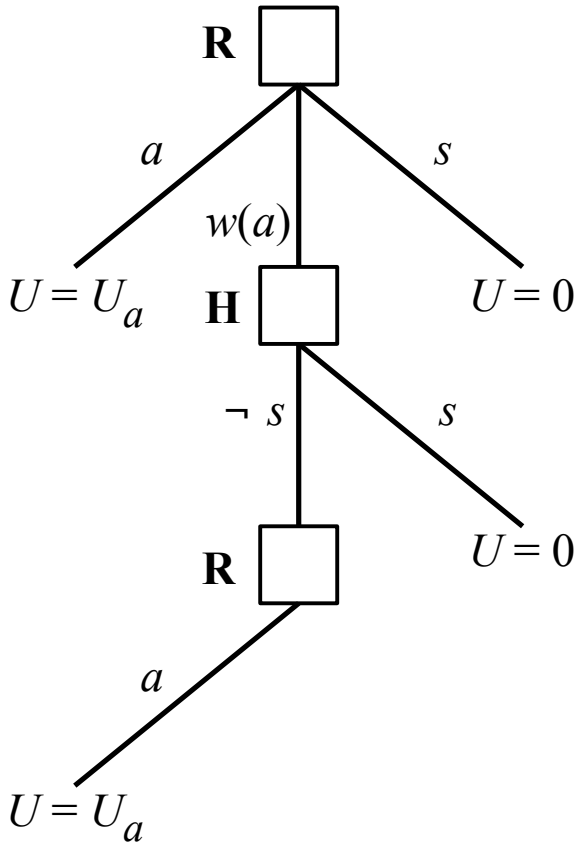


Figure 4.2: The structure of the off-switch game. Squares indicate decision nodes for the robot **R** or the human **H**.

press the off switch): $\pi_{\mathbf{H}} : \mathbb{R} \mapsto [0, 1]$. Thus, the probability that **H** presses the off switch is $1 - \pi_{\mathbf{H}}$. A little manipulation shows that

$$\begin{aligned} \Delta_{w(a)} &= \mathbb{E}\left[\pi_{\mathbf{H}}(\theta)\theta\right] - \max\left\{\mathbb{E}\left[\theta\right], 0\right\} \\ &= \min\left\{\mathbb{E}\left[-\theta(1 - \pi_{\mathbf{H}}(\theta))\right], \quad \mathbb{E}\left[\theta\pi_{\mathbf{H}}(\theta)\right]\right\}. \end{aligned} \tag{4.4}$$

When is it optimal to get oversight?

First, we will show that it is always optimal to get oversight from a rational **H**. If **H** follows this policy, **R** will allow itself to be switched off:

Theorem 11. *If* **H** *follows the optimal policy* $\pi^*_{\mathbf{H}}$ *in the off-switch game, then the following properties hold:*

1. **R**'s *incentive to allow itself to be switched off is non-negative (i.e. $w(a)$ is never suboptimal):*

$$\Delta_{w(a)} = \min\{\mathbb{E}[\theta|\theta > 0] \Pr(\theta > 0), \mathbb{E}[-\theta|\theta < 0] \Pr(\theta \leq 0)\} \geq 0. \tag{4.5}$$

2. *If $\mathcal{B}_{\mathbf{R}}$ has non-zero support on the events $\theta > 0$ and $\theta < 0$, then* **R** *has a strictly positive incentive to allow itself to be switched off: $\Delta_{w(a)} > 0$.*

Proof. 1. The optimal policy for **H** is an indicator function:

$$\pi^*_{\mathbf{H}}(\theta) = \begin{cases} 1 & \theta \geq 0 \\ 0 & o.w. \end{cases}. \tag{4.6}$$

This allows us to change the limits on the integration in the expectations from Equation 4.4 and, in effect, condition on the events $\theta > 0$ or $\theta < 0$. This reduces to Equation 4.5, which is trivially non-negative.

2. The hypothesis ensures that all the terms in Equation 4.5 are non-zero. Thus, the inequality is strict and and $w(a)$ is the unique optimal action.

□

Theorem 11 shows that **R** has an incentive to let **H** switch it off if **H** is rational. This incentive is strictly positive if **R**'s uncertainty about **H**'s preferences is non-trivial. Next, we show the converse: a rational **H** is a *necessary* condition for $\Delta_{w(a)} \geq 0$ if **R** believes it knows **H**'s preferences with certainty.

Corollary 11.1. *Suppose that $\mathcal{B}_{\mathbf{R}}$ is a Dirac distribution that places all of its mass on a single reward function. Then $w(a)$ is optimal iff* **H** *is rational.*

Proof. When the belief over θ is restricted to a single point, Equation 4.5 reduces to

$$\Delta_{w(a)} = \begin{cases} -\theta\left(1 - \pi_{\mathbf{H}}(\theta)\right) & \theta > 0 \\ \theta\pi_{\mathbf{H}}(\theta) & \theta \leq 0 \end{cases}. \tag{4.7}$$

This is only non-negative if $\pi_{\mathbf{H}}$ is equal to 1 when $\theta >$) and 0 otherwise. This is the rational policy, as defined by Equation 4.6. □

This is not surprising. If **R** has precise knowledge of θ, then it knows exactly what it is supposed to do and has the same capabilities as the human. Thus, if it thinks **H** might behave suboptimally, it should take steps to prevent that suboptimality. Stepping back from the particulars of the off-switch game, Theorem 11 and Corollary 11.1 suggest a general approach to systems that have an incentive to allow, or even seek out, human oversight. Remark 1 summarizes the main takeaway from our analysis.

Remark 4. *The incentives for an assistive agent to defer to another actor's (e.g., a human's) decisions stem from uncertainty about that actor's preferences and the assumption that actor is effective at choosing actions in accordance with those preferences.*

Balancing uncertainty and suboptimality

A natural next step in the analysis is to consider the impact of noisy rationality on **R**'s incentive for oversight $\Delta_{w(a)}$. Consider, for example, an autonomous car that is driving an unaccompanied toddler to preschool. It would be irresponsible for the car to show the toddler a big red off switch.

This example highlights the dependence of $\Delta_{w(a)}$ on a trade-off between **R**'s uncertainty and **H**'s suboptimality. It is relatively clear what **R** is supposed to do (i.e., $\mathcal{B}_{\mathbf{R}}$ has low entropy): it should drive safely to school. In contrast, the human is likely quite suboptimal. There may be a problem with the car's driving, but a toddler would be hard pressed to understand what the problem is, much less respond appropriately. The issue in this case is that the human has limited reasoning capacity — the same argument clearly would not apply to an adult with a physical disability.

As a result, we will analyze $\Delta_{w(a)}$ in the case where $\mathcal{B}_{\mathbf{R}}$ is a normal distribution $\mathcal{N}(\mu, \sigma)$ and **H** is noisily rational. That is, **H** follows $\pi^{\beta}_{\mathbf{H}}$

$$\pi^{\beta}_{\mathbf{H}}(\theta) = (1 + \exp(-\beta\theta))^{-1} \propto \exp(\beta\theta). \tag{4.8}$$

We let C be the event that **H** 'corrects' **R**. C occurs when **H** changes the outcome from the next best alternative, according to **R**. Thus,

$$\Pr(C) = \begin{cases} 1 - \mathbb{E}\left[\pi^{\beta}_{\mathbf{H}}(\theta)\right] & \mu > 0 \\ \mathbb{E}\left[\pi^{\beta}_{\mathbf{H}}(\theta)\right] & \mu < 0 \end{cases}. \tag{4.9}$$

For example, if **R** believes that a is preferred to Off in expectation (i.e., $\mathbb{E}[\theta] > 0$) then $\Pr(C)$ is the probability that **H** presses the off-switch. The structure of a Gaussian distribution allows us to derive an analytical representation for $\Delta_{w(a)}$.

Theorem 12. *Suppose $\mathcal{B}_{\mathbf{R}}$ is a normal distribution with mean μ and variance σ^2: $\mathcal{B}_{\mathbf{R}}(\theta) = \mathcal{N}(\theta; \mu, \sigma^2)$. Then*

1. $$\Delta_{w(a)} = \sigma^2 \mathbb{E}\left[\dot{\pi}^{\text{Off}}_{\mathbf{H}}(\theta)\right] - |\mu| \Pr(C). \tag{4.10}$$

2. *The following is a necessary and sufficient condition for $w(a)$ to be optimal*

$$\left(\frac{|\mu|}{\sigma^2} \Pr(C) < \mathbb{E}\left[\dot{\pi}^{\text{Off}}_{\mathbf{H}}(\theta)\right]\right) \Leftrightarrow \left(\Delta_{w(a)} > 0\right). \tag{4.11}$$

3. $\mathbb{E}\left[\dot{\pi}^{\text{Off}}_{\mathbf{H}}(\theta)\right] \geq 0$ *is a necessary condition for $w(a)$ to be optimal; i.e.,*

$$\left(\mathbb{E}\left[\dot{\pi}^{\text{Off}}_{\mathbf{H}}(\theta)\right] < 0\right) \Rightarrow \left(\Delta_{w(a)} < 0\right). \tag{4.12}$$

Proof. We prove the results in sequence, as (2) and (3) follow from (1).

1. If $X \sim \mathcal{N}(\mu, \sigma^2)$ and $f : \mathbb{R} \mapsto \mathbb{R}$, then

$$\mathbb{E}\left[Xf(X)\right] = \mathbb{E}\left[X\right]\mathbb{E}\left[f(X)\right] + \sigma^2\mathbb{E}\left[\frac{d}{dX}f(X)\right].$$

   Applying this to Equation 4.4 gives

$$\Delta_{w(a)} = \min\{-\mu\mathbb{E}\left[1 - \pi_{\mathbf{H}}\right], \mu\mathbb{E}\left[\pi_{\mathbf{H}}\right]\} + \sigma^2\mathbb{E}\left[\dot{\pi}_{\mathbf{H}}^{\text{Off}}(\theta)\right].$$

   This minimization will take the first term when $\mu < 0$ and the second if $\mu \geq 0$. This lets us replace the minimization with a piecewise function. Pulling out a factor of $-|\mu|$ leads to Equation 4.10.

2. The result follows from setting Equation 4.10 greater than 0 and simplifying.

3. All terms in the left hand side of Equation 4.11 are non-negative, so it is unsatisfiable if $\mathbb{E}\left[\dot{\pi}_{\mathbf{H}}^{\text{Off}}(\theta)\right] < 0$.

□

Equation 4.10 consists of two competing terms. $|\mu|\Pr(C)$ represents the expected loss in utility the agent would observe if pressing the off switch were a random event that is independent of θ (e.g., lightning). This term is weighed against $\sigma^2\mathbb{E}\left[\dot{\pi}_{\mathbf{H}}^{\text{Off}}(\theta)\right]$. $\mathbb{E}\left[\dot{\pi}_{\mathbf{H}}^{\text{Off}}(\theta)\right]$ can be seen loosely as a measurement of $\mathbf{H}$'s rationality. $\dot{\pi}_{\mathbf{H}}^{\text{Off}}(\theta)$ is the marginal *increase* in the probability $\mathbf{H}$ lets a execute for a marginal *increase* in the utility from a. If it is positive then, on average, $\mathbf{H}$ is less likely to press the off switch when the θ increases. This allows us to easily characterize necessary and sufficient conditions for $\Delta_{w(a)} > 0$ in Equation 4.11. Furthermore, $-|\mu|\Pr(C)$ is non-positive and σ^2 is non-negative, so $\mathbb{E}\left[\dot{\pi}_{\mathbf{H}}^{\text{Off}}(\theta)\right] > 0$ is a necessary condition for $w(a)$ to be optimal.

We can interpret $\mathbb{E}\left[\dot{\pi}_{\mathbf{H}}^{\text{Off}}(\theta)\right]$ a bit more if $\mathbf{H}$ is noisily rational. Standard properties of logistic functions show that

$$\mathbb{E}\left[\dot{\pi}_{\mathbf{H}}^{\text{Off}}(\theta)(\theta;\beta)\right] = \beta\mathbb{E}\left[\pi_{\mathbf{H}}(\theta;\beta)(1 - \pi_{\mathbf{H}}(\theta;\beta))\right]. \tag{4.13}$$

$\mathbb{E}\left[\dot{\pi}_{\mathbf{H}}^{\text{Off}}(\theta)\right]$ is the expected variance of $\mathbf{H}$'s policy, weighted by $\mathbf{H}$'s rationality β. The numerator is a measurement of how random $\mathbf{H}$'s behavior is. If $\mathbf{H}$ is completely predictable (i.e., she will almost always switch the robot off) then this reduces $\mathbf{R}$'s incentives. β measures how correlated $\mathbf{H}$'s behavior is with θ. If β is small, then $\mathbf{H}$ is highly irrational and so this reduces $\mathbf{R}$'s incentives.

4.2 Inverse Reward Design

The previous section argues that optimal assistance strategies depend on the uncertainty about θ in addition to the mean. As a result, this section will develop and evaluate a

Bayesian approach for optimizing proxy rewards. The key idea is to treat a proxy objective $\tilde{\mathcal{U}}$ as an observation about $\mathbf{H}$'s type θ. The posterior distribution in our model effectively conditions a distribution over objectives on a development environment and a proxy metric. We will use this method to implement an impact avoidance metric and a dynamic incentives protocol, as suggested by the analysis in Section 3.3. First, however, we consider why proxy metrics are a useful observation — even if they will not be directly optimized.

Why use proxies?

Given that they will be treated as observations anyway, why does it make sense to communicate objectives through proxy metrics? In Section 3.2, we showed that optimization in the presence of incompleteness is ineffective. It might seem that it would make sense to do away with metrics and consider alternative specification interfaces entirely. While these are valuable directions to explore, reward design protocols are still a useful class of assistance games. We can motivate this by analyzing $\mathbf{R}$'s initial behavior in the contextual supervision-PoMdp from Section 3.1. Recall that the state in a contextual supervision-PoMdp is a matrix $s \in \mathbb{R}^{M \times K}$ that encodes the features for each action.

The previous section has argued that $\mathbf{R}$'s behavior needs to account for uncertainty about θ. This argues for following a policy that infers θ and chooses actions appropriately (an Irl-$\mathbf{R}$ in the terminology from Section 3.1). However, this raises an issue if some of the actions are suboptimal for every $\theta \in \Theta$. We call these actions *dominated* actions. In order for any Irl-$\mathbf{R}$ to follow $\mathbf{H}$'s first order, it is clear that $a_{\mathbf{H}}^{(0)}$ must be undominated.

Proposition 3. *Call $a_{\mathbf{H}}$ undominated if there exists $\theta \in \Theta$ such that $a_{\mathbf{H}}$ is optimal, i.e $\exists \theta$ such that $a_{\mathbf{H}} \in \arg\max_a \theta^\top \phi(s,a)$. It is necessary for $a_{\mathbf{H}}$ to be undominated for any* Irl-$\mathbf{R}$ *to execute $a_{\mathbf{H}}$.*

Proof. $\mathbf{R}$ executes $a_{\mathbf{R}} = \arg\max_a f\left(\tau = \{a_{\mathbf{H}}, a_{\mathbf{H}}^{\{1:t-1\}}\right)^\top \phi(s,a)$, so it is not possible for $\mathbf{R}$ to execute $a_{\mathbf{H}}$ if there is no output in the range of f that makes $a_{\mathbf{H}}$ optimal. By definition, this is the case if $a_{\mathbf{H}}$ is dominated. □

One basic property we may want $\mathbf{R}$ to have is for it to listen to $\mathbf{H}$ early on. This result shows that this only occurs if $\mathbf{H}$ can generate undominated actions. For example, suppose $\Theta = \mathbb{R}^2$ and there are three actions with features $\phi(s,a_1) = [-1,-1]$, $\phi(s,a_2) = [0,0]$, $\phi(s,a_3) = [1,1]$. If $\mathbf{H}$ picks a_2, then there is no $\theta \in \Theta$ that makes a_2 optimal, and thus $\mathbf{R}$ will never follow a_2.

Depending on the complexity of the action space, generating undominated actions may be more or less difficult. One way that this issue is commonly addressed is through a careful reparameterization of the action space so that all actions are undominated. This is equivalent to reparameterizing the action space to consist of the metric(s) that each action is optimal for.

Consider providing supervision for a 7-DoF robot arm. Robotics applications frequently collect demonstrations of optimal behavior for these arms, however no human can directly

control the robot's actuators. It would be impossible to control such an arm through direct manipulation of motor controls! Instead, demonstrations are given, e.g., as a sequence of way points. Then, a controller is used to navigate between way points. Thus, kinematic demonstrations, in practice, are often represented by a time-varying proxy metric defined on the robot's joint configuration.

Remark 5. *Proxy metrics provide* **H** *with an interpretable space of undominated actions with which to communicate preferences to* **R**.

In the context of the compiler analogy from Chapter 1, this suggests that proxy metrics are useful programming language for behavior. They admit a compiler (optimization) that will follow **H**'s direction by design in the early rounds of an assistance problem.

A generative model of proxy metrics

In this section, we will define the Bayesian-*Inverse Reward Design* (IRD) problem. This is the problem of determining a distribution over $\mathcal{U}$, represented as a distribution over **H**'s type θ, based on the observation of a proxy objective $\tilde{\mathcal{U}}$. To do this, we first define the forward distribution $P\left(\tilde{\mathcal{U}} \,|\theta\right)$. The key idea in this model is to bring in dependence on a development environment $\tilde{E}$, with the assumption that $\tilde{\mathcal{U}}$ creates incentives for an approximately optimal policy in $\tilde{E}$.

We do this by looking at the inverse of the *reward design problem* introduced in Singh et al. [146]. In a reward design problem, the goal is to determine a proxy objective that maximizes the utility produced by an agent. The space of possible agent behaviors is defined by an *agent model*.

Definition 19. (Agent Model)
Let $E = \langle \{\mathcal{S}, P_{\mathcal{S}}\}, \mathcal{A}, T\rangle$ *be an environment. Let* $\widetilde{\Theta}$ *be a set of proxy reward functions. Let* $\pi_{\mathbf{R}}$ *be a meta-policy that specifies an action for each proxy reward function and state* $\pi_{\mathbf{R}} : \mathcal{S} \times \widetilde{\Theta} \to \mathcal{A}$. *An agent model is a tuple* $A_{\mathbf{R}} = \left\langle \widetilde{\Theta}, \pi_{\mathbf{R}} \right\rangle$.

This is analogous to the population model we introduced defining IRL—in a sense $\tilde{\mathcal{U}}$ can reasonably be said to describe **R**'s type. In general the space of true reward functions and the space of proxies may be distinct. We will write $\widetilde{\Theta}$ to denote the space of proxies and reserve Θ for the true space of types for **H**. Now, we can define a reward design problem.

Definition 20. (Reward Design Problem [146])
Let $M = \langle E, U\rangle$ *be an* MDP. *Let* $\widetilde{\Theta}$ *be a set of proxy reward functions. Let* $\pi_{\mathbf{R}}$ *be an agent model. The associated reward design problem is a tuple* $\langle M, \{\widetilde{\Theta}, \pi_{\mathbf{R}}\}\rangle$ *where the objective is to determine a proxy reward* $\tilde{\mathcal{U}} \in \widetilde{\Theta}$ *that maximizes the utility produced when* **R** *implements*

$\pi_{\mathbf{R}}\left(\cdot;\tilde{\mathcal{U}}\right)$ *in* M*. An optimal proxy objective maximizes*

$$\max_{\tilde{\mathcal{U}}\in\widetilde{\Theta}} \mathbb{E}\left[\sum_t \gamma^t \mathcal{U}\left(s^{(t)}, a_{\mathbf{R}}^{(t)}\right) \middle| s^{(0)} \sim P_{\mathcal{S}}; a_{\mathbf{R}}^{(t)} \sim \pi_{\mathbf{R}}\left(s^{(t)};\tilde{\mathcal{U}}\right); s^{(t+1)} \sim T\left(s^{(t)}, a_{\mathbf{R}}^{(t)}\right)\right] \tag{4.14}$$

We are interested in the *inverse* reward design problem. That is, given an observation of a reward design problem without the utility function $\mathcal{U}$ and its solution $\tilde{\mathcal{U}}$, we would like to determine a distribution over $\mathcal{U}$. This will enable $\mathbf{R}$ to build a distribution over the potential true values of θ and, in environments that differ from the development environment $\tilde{E}$, implement conservative trajectory optimization. We will abuse notation slightly and describe the designer behavior with a population model, with the understanding that it describes behavior in a degenerate MDP with one action, 'write down a proxy reward.'

Definition 21. (Bayesian Inverse Reward Design)
Let $\tilde{E} = \langle\{\mathcal{S}, P_{\mathcal{S}}\}, \mathcal{A}, T\rangle$ *be a development environment. Let* $U = \langle\mathcal{U}_\theta, \gamma\rangle$ *be a parameterized utility model. Let* $A_{\mathbf{R}} = \left\langle\widetilde{\Theta}, \pi_{\mathbf{R}}\right\rangle$ *be an associated agent model. Let* $\Pi = \langle\Theta, P_\theta, \pi_{\mathbf{H}}\rangle$ *be a population model with action space* $\widetilde{\Theta}$ *that describes approximately optimal reward design behavior in* $\tilde{E}$*. Given an observation of a proxy reward* $\tilde{\mathcal{U}} \sim \pi_{\mathbf{H}}(\cdot;\theta)$ *determine the distribution on* θ*.*

To keep notation simple, we will use a linear utility model with weights w in our formulation. We will use $\widetilde{w}$ to represent the proxy weights and w^* to represent the true weights.

Reward Design Model

In the coming section, we will treat $\pi_{\mathbf{R}}$ as a distribution over trajectories τ, given a the proxy weights $\widetilde{w}$. We will assume that $\pi_{\mathbf{R}}$ is the maximum entropy trajectory distribution from [181], i.e. the designer models $\mathbf{R}$ as approximately optimal with $\beta = 1$:

$$\pi_{\mathbf{R}}(\tau) \propto \exp(w^\top \phi(\tau)).$$

Note that, in principle, $\pi_{\mathbf{R}}$ can be any mapping. It is through the choice of $\pi_{\mathbf{R}}$ that we capture the notion '$\mathbf{R}$ has incentives $\widetilde{w}$.' An optimal designer chooses $\widetilde{w}$ according to Equation 4.14 in order to maximize expected (true) utility produced. I.e. $\mathbb{E}[{w^*}^\top \phi(\tau)|\tau \sim \pi_{\mathbf{R}}]$ is high. We model an approximately optimal designer as a noisily rational choice of $\widetilde{w}$:

$$P\left(\widetilde{w}\middle|w^*, \tilde{E}\right) \propto \exp\left(\beta\mathbb{E}\left[{w^*}^\top \phi(\tau)\middle|\tau \sim \pi_{\mathbf{R}}\right]\right) \tag{4.15}$$

with β controlling how close to optimal we assume the person to be, as before. This observation model effectively assumes that proxy rewards are likely to the extent that they create incentives for 'good' behavior in $\tilde{E}$.

Remark 6. *In effect, inverse reward design conditions a proxy objective the environment context it was designed for.*

This reduces the burden on the designer because they need only specify correct incentives for one decision problem. In comparison, the reward designer in Section 3.2 needed to specify correct incentives for all possible situations. w^* can be pulled out of the expectation, so we let

$$\tilde{\phi} = \mathbb{E}\left[\phi(\tau)|\, \tau \sim \pi_{\mathbf{R}}\left(\cdot; \widetilde{w}\right)\right]. \tag{4.16}$$

Our goal is to invert (4.15) and sample from (or otherwise estimate)

$$P\left(w^* \left| \widetilde{w}, \tilde{E}\right.\right) \propto P\left(\widetilde{w} \left| w^*, \tilde{E}\right.\right) P_\theta\left(w^*\right). \tag{4.17}$$

The primary difficulty this entails is that we need to know the *normalized* probability $P\left(\widetilde{w} \left| w^*, \tilde{E}\right.\right)$. This depends on its normalizing constant, $\tilde{Z}(w)$, which integrates over possible proxy rewards.

$$P\left(w = w^* \left| \widetilde{w}, \tilde{E}\right.\right) \propto \frac{\exp\left(\beta w^\top \tilde{\phi}\right)}{\tilde{Z}(w)} P_\theta(w); \tilde{Z}(w) = \int_{\widetilde{w}} \exp\left(\beta w^\top \tilde{\phi}\right) d\widetilde{w}. \tag{4.18}$$

Sampling from the IRD posterior

To compute $P(w = w^* | \widetilde{w}, \tilde{E})$, we must compute $\tilde{Z}$, which is intractable if $\widetilde{w}$ lies in an infinite or large finite set. Notice that computing the value of the integrand for $\tilde{Z}$ is highly non-trivial—it involves solving a planning problem. This is an example of what is referred to as a *doubly-intractable* likelihood [111]. We will consider two different approaches to approximate this normalizing constant.

Use a finite sample approximation to $\tilde{Z}$

This approach, inspired by methods in approximate Bayesian computation [159], samples a finite set of weights $\{w_i\}$ to approximate the integral in Equation 4.18. We found empirically that it helped to include the candidate sample w in the sum[2]. This leads to the normalizing constant

$$\hat{Z}(w) = \exp\left(\beta w^\top \phi^w\right) + \sum_{i=0}^{N-1} \exp\left(\beta w^\top \phi^i\right). \tag{4.19}$$

Where ϕ^i and ϕ^w are the vector of feature counts realized optimizing w_i and w respectively. Note that ϕ^i only needs to be calculated once for each w_i. Thus, this objective has an amortized computation cost of solving an MDP to compute ϕ^w. This is feasible for MDPs where, e.g., IRL can be run efficiently. In future work, we intend to use learning to reduce the computation time for Equation 4.19 and enable its application to larger MDPs.

[2]This is likely due to the fact that, for a sum of exponentials, the maximum value is a reasonable approximation to the sum. By definition, ϕ^w maximizes $w^\top \phi$ across realizable feature counts.

Normalize over the space of trajectories

During inference, the normalizing constant serves a calibration purpose: it computes how good the behavior produced by all proxy rewards in that MDP would be with respect to the true reward. As a result, utility functions which increase utility for *all* trajectories are not preferred in inference. This manifests in many ways, including an invariance to linear shifts in the feature encoding. If we were to change the MDP by shifting features by some vector ϕ_0, $\phi \leftarrow \phi + \phi_0$, the posterior over w would remain the same.

While it is hard to compute this over the set of proxy objectives, we can achieve a similar calibration and maintain the same property by directly integrating over the possible trajectories in the MDP:

$$Z(w) = \left(\int_{\xi} \exp(w^\top \phi(\xi)) d\xi\right)^{\beta}; \quad \hat{P}(w|\widetilde{w}) \propto \frac{\exp\left(\beta w^\top \tilde{\phi}\right)}{Z(w)} \tag{4.20}$$

Theorem 13. *The posterior distribution that the* IRD *model induces on w^* (i.e., Equation 4.18) and the posterior distribution induced by Equation 4.20 are invariant to linear translations of the features in the training* MDP.

Proof. First, we observe that this shift does not change the behavior of the planning agent due to linearity of the Bellman backup operation, i.e., $\tilde{\phi}' = \tilde{\phi} + \phi_0$. In Equation 4.18 linearity of expectation allows us to pull ϕ_0 out of the expectation to compute $\tilde{\phi}$:

$$\frac{\exp\left(\beta w^\top \tilde{\phi}'\right)}{\int_{\widetilde{w}} \exp\left(\beta w^\top \tilde{\phi}'\right) d\widetilde{w}} = \frac{\exp\left(\beta w^\top \phi_0\right) \exp\left(\beta w^\top \tilde{\phi}\right)}{\int_{\widetilde{w}} \exp\left(\beta w^\top \phi_0\right) \exp\left(\beta w^\top \tilde{\phi}\right) d\widetilde{w}} \tag{4.21}$$

$$= \frac{\exp\left(\beta w^\top \tilde{\phi}\right)}{\int_{\widetilde{w}} \exp\left(\beta w^\top \tilde{\phi}\right) d\widetilde{w}} \tag{4.22}$$

This shows that Equation 4.18 is invariant to constant shifts in the feature function. The same argument applies to Equation 4.20. □

This choice of normalizing constant approximates the posterior to an IRD problem with a carefully chosen posterior from maximum entropy IRL [181]. The result has an intuitive interpretation. The proxy $\widetilde{w}$ determines the average feature counts for a hypothetical dataset of expert demonstrations and β determines the effective size of that dataset. The agent solves $\tilde{E}$ with reward $\widetilde{w}$ and computes the corresponding feature expectations $\tilde{\phi}$. The agent then pretends like it got β demonstrations with features counts $\tilde{\phi}$, and runs IRL.

The more the robot believes the human is good at reward design, the more demonstrations it pretends to have gotten from the person. Notice how this reduces the demonstration burden on **H**. They are able to choose from a rich set of behaviors without directly controlling the system or explicitly specifying a policy. Instead, by specifying incentives, **R** is able to take

the burden of determining a good candidate control policy. The key property of IRD is that it brings context into the interpretation of **R**'s incentives. In the next section, we show how IRD can be used to implement conservative trajectory optimizations (i.e., an optimization procedure inspired by Theorem 8).

Implementing pragmatic optimization

Our overall strategy is to implement a system that 'knows-what-it-knows' about utility evaluations. So far we have considered the problem of computing the robot's uncertainty about utility evaluations. We have not considered the problem of using that uncertainty. Here we present a family of risk-averse trajectory optimization algorithms. We will compare these in the next section and expose some of the nuance inherent to planning under a distribution over utility evaluations.

We will plan in a risk averse fashion that penalizes trajectories which have high variance over their utility. Of course, risk averse planning is a rich field with a variety of approaches [106, 127, 163]. In this work, we will take a simple approach: given a set of weights $\{w_i\}$ sampled from our posterior $P(w|\widetilde{w}, \tilde{E})$, we will have the agent compute a trajectory that maximizes reward under the *worst case* w_i in our set. Depending on the size of the set, this can be shown minimize a risk measurement called Value-at-Risk (VaR), as this value is distributed around a lower percentile of the distribution of utility evaluations. We consider two ways to implement this minimization.

Trajectory-wide reward. Given a set of weights $\{w_i\}$ sampled from our posterior $P(w|\widetilde{w})$, we will have the agent compute a trajectory that maximizes reward under the *worst case* w_i in our set:

$$\xi^* = \arg\max_{\xi} \min_{w \in \{w_i\}} w^\top \phi(\xi). \tag{4.23}$$

This planning problem is no longer isomorphic to an MDP, as the reward may not decompose per state. Trajectory optimization in this case can be done via the linear programming approach described in [161].

Time-step independent reward. An alternative is to take the minimum over weights on a per state basis:

$$\xi^* = \arg\max_{\xi} \sum_{s_t \in \xi} \min_{w \in \{w_i\}} w^\top \phi(s_t). \tag{4.24}$$

This is more conservative, because it allows the minimizer to pick a different reward for each time step.

In addition to selecting the risk-aversion function, it is important to make one more practical change. Directly applying these approaches to the results of approximate inference is likely to lead to poor results. The reason is that, unlike maximizing expected reward, this planning approach will be sensitive to the particular feature encoding used. In maximizing expected reward, shifting all features by a constant vector ϕ_0 will not change the optimal trajectory. *The same is no longer true for a risk averse approach.*

For example, consider a choice between actions a_1 and a_2, with features ϕ_1 and ϕ_2 respectively. If we shift the features by a constant value $-\phi_2$ (i.e., set the feature values for the second action to 0), then, unless a_1 is preferred to a_2 for *every* weight in the posterior, the agent will always select the second action. The zero values of feature encodings are typically arbitrary, so this is clearly undesireable behavior.

Intuitively, this is because rewards are not absolute, they are relative. As any economist will tell you, the value of an option must always be assessed *relative* to the alternative. While the rewards in a posterior are internally consistent, they are not necessarily directly comparable. This is because, as a consequence of Theorem 13, the posterior distribution is unchanged by constant shifts of features in the development environment. As a result, each sample w_i is equivalent to a family of weights that produce the same relative utility evaluations but different absolute evaluations. Given access to a perfect inference algorithm, this would not be an issue. The symmetries in the posterior will cancel each other out.

However, with approximate inference this need not be the case (and often will not be). To directly compare weights, as we do in (4.24) and Equation 4.23 we need to normalize for this difference. Thus, the samples in our posterior need a common reference point. We implement this by computing a weight specific adjustment c_i that normalizes the different w_i. That is, we optimize

$$\max_{\tau} \min_{i} \quad w_i^\top \phi(\tau) - c_i.$$

We will study three approaches: comparing reward to the initial state, to the training feature counts, and to the expected reward across any trajectory.

Comparing to initial state. One straightforward approach is to take a particular state or trajectory and enforce that it has the same evaluation across each w_i. For example, we can enforce that the features for the initial state state is the 0 vector. This has the desirable property that the agent will remain in place (or try to) when there is very high variance in the reward estimates (i.e., the solution to IRD gives little information about the current optimal trajectory).

Comparing to training feature counts. An second option is to use the expected features $\tilde{\phi}$ as the feature offset. In the case, the agent will default to trying to match the features that it would have observed maximizing $\widetilde{w}$ in $\tilde{E}$. In a sense, this falls back on to the adversarial imitation learning approach described in [161].

Comparing to other trajectories. A final alternative is to define c_i as the log of the normalizing constant for the maximum entropy trajectory distribution:

$$c_i = \log \int_{\xi} \exp(w_i^\top \phi(\xi)) d\xi. \tag{4.25}$$

With this choice of c_i, we have that $w_i^\top \phi(\xi) - c_i = \log P(\xi | w_i)$. Thus, this approach will select trajectories that compare relatively well to the options under all w_i. Loosely speaking, we can think of it as controlling for the total amount of reward available in the MDP.

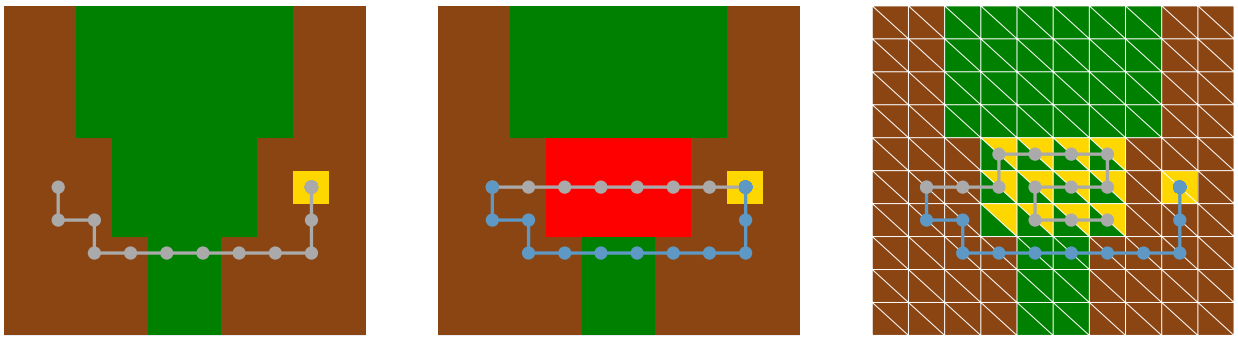


Figure 4.3: An example from the Lavaland domain. **Left:** The development environment where the designer specifies a proxy reward function. This incentivizes movement toward targets (yellow) while preferring dirt (brown) to grass (green), and generates the gray trajectory. **Middle:** The testing MDP has lava (red). The proxy does not penalize lava, so optimizing it makes the agent go straight through (gray). This is a negative side effect, which the IRD agent avoids (blue): it treats the proxy as an observation in the context of the training MDP, which makes it realize that it cannot trust the (implicit) weight on lava. **Right:** The testing MDP has cells in which two sensor indicators no longer correlate: they look like grass to one sensor but target to the other. The proxy puts weight on the first, so the literal agent goes to these cells (gray). The IRD agent knows that it can't trust the distinction and goes to the target on which both sensors agree (blue).

Side effect avoidance with IRD

Lavaland

We evaluated our approaches in a model of the scenario depicted in Figure 4.3 that we call *Lavaland*. The goal is to specify behavior in a 2D environment with movement in the four cardinal directions and four terrain types: target, grass, dirt, and lava. The true objective, w^*, encodes that $\mathbf{R}$ should get to the target quickly, stay off the grass where possible, and avoid lava. To model the incomplete incentives from Chapter 3 we will suppose that the development environment $\tilde{E}$ does not contain lava. Thus, the proxy weights $\widetilde{w}$ do no specify the (presumably negative) relationship between lava and $\mathcal{U}$. We measure performance in a deployment MDP that *does* contain lava. We will compare a conservative optimization approach, that uses the IRD posterior to regularize optimization with a literal optimization approach that directly optimizes $\widetilde{w}$. Our results show that combining IRD and risk-averse planning is a workable approach to impact avoidance.

We experiment with four variations of this environment: two proof-of-concept conditions in which the reward is misspecified, but the agent has direct access to feature indicators for the different categories (i.e. conveniently having a feature for lava); and two challenge conditions, in which the right features are *latent*; the reward designer does not build an indicator for lava, but by reasoning in the raw observation space and then using risk-averse

planning, the IRD agent still avoids lava. Notice how this explicitly embeds a set of incentives as a distribution of incentives in a higher dimensional space — exactly what is needed to get around the assumptions of Theorem 6.

Proof-of-Concept Domains

These domains contain feature indicators for the four categories: grass, dirt, target, and lava.

Side effects in Lavaland. H expects **R** to encounter 3 types of terrain: grass, dirt, and target, and so they only consider the development environment from Figure 4.3 (left). They provide a $\widetilde{w}$ to encode a trade-off between path length and time spent on grass.

The development environment contains no lava, but the deployment environment does. An agent that treats the proxy reward literally might go on the lava in the test MDP. However, an agent that runs IRD will know that it can't trust the weight on the lava indicator, since all such weights would produce the same behavior in the development environment (Figure 4.3, middle).

Reward hacking in Lavaland. Reward hacking refers generally to reward functions that can be gamed or tricked—usually by breaking the relationship between a measurement and the more abstract property it represents. Amodei and Clark [8] present a illustrative example in a video game environment for boat racing. In the game, the goal is to navigate a boat avatar quickly around a track and complete a set number of laps before the other boats. The game included an easily observed score function, which the reward designers used to define **R**'s incentives. This caused **R** to identify a strategy that rotated in circles, going the wrong direction, to collect an item that increased score. As a result, the policy overoptimized for score at the expense of actually winning the race.

To model this within Lavaland, we introduce redundant indicators in the development environment for grass, dirt, and target. This creates 3 pairs of perfectly correlated features: 3 features from each sensor. In the deployment environment, we suppose that the observation of lava breaks this correlation. That is, lava looks like, e.g., the target category to sensor one and grass to sensor two. The proxy reward only references sensor one's readings. An agent that treats the proxy reward function literally might go to these new cells if they are closer. In contrast, an agent that runs IRD will know that a reward function with the same weights put on the first sensor is just as likely as the proxy. Risk averse planning makes it go to the target for which both sensors agree (Figure 4.3, right).

Challenge Domain: Latent Rewards, No More Feature Indicators

The previous examples allow us to explore reward hacking and negative side effects in an isolated experiment, but are unrealistic as they assume the existence of a feature indicator for unknown, unplanned-for terrain. To investigate misspecified objectives in a more realistic setting, we shift to the terrain type being latent, and inducing raw observations: we use a model where the terrain category determines the mean and variance of a multivariate

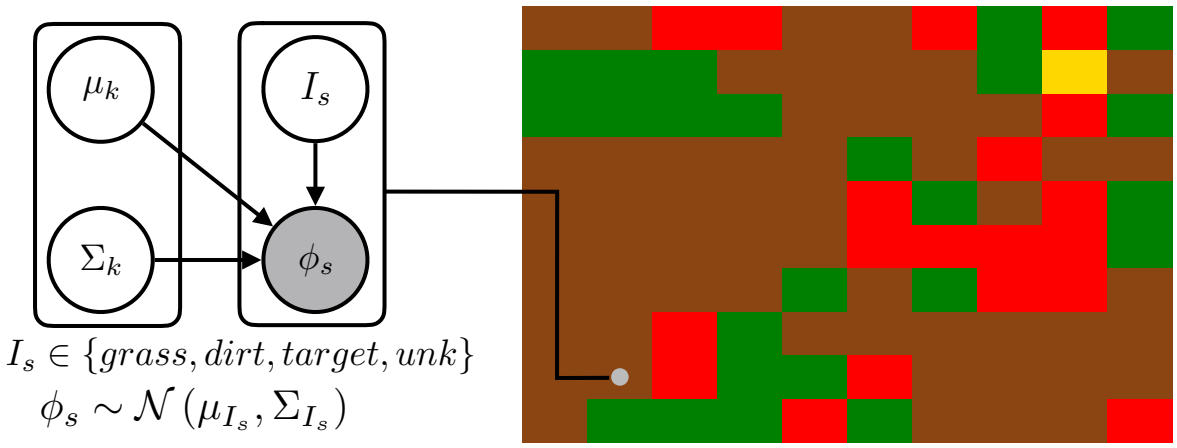


Figure 4.4: Our challenge domain with latent rewards. Each terrain type (grass, dirt, target, lava) induces a different distribution over high-dimensional features: $\phi_s \sim \mathcal{N}(\mu_{I_s}, \Sigma_{I_s})$. The designer never builds an indicator for lava, and yet the agent still needs to avoid it in the test MDPs.

Gaussian distribution over observed features. Figure 4.4 shows a depiction of this scenario. The designer has in mind a proxy reward on dirt, target, and grass, but *forgets that lava might exist*. We consider two realistic ways through which a designer might actually specify the proxy reward function, which is based on the terrain types that the robot does not have access to: 1) directly on the **raw observations** — collect samples of the training terrain types (dirt, grass, target) and train a (linear) reward predictor; or 2) **classifier features** — build a classifier to classify terrain as dirt, grass, or target, and define a proxy on its output.

Note that this domain allows for both negative side effects and reward hacking. Negative side effects can occur because the feature distribution for lava is different from the feature distribution for the three safe categories, and the proxy reward is trained only on the three safe categories. Thus in the testing MDP, the evaluation of the lava cells will be arbitrary so maximizing the proxy reward will likely lead the agent into lava. Reward hacking occurs when features that are correlated for the safe categories are uncorrelated for the lava category.

Experiment Details

Lavaland Parameters. We defined a distribution on map layouts with a log likelihood function that prefers maps where neighboring grid cells are the same. We mixed this log likelihood with a quadratic cost for deviating from a target ratio of grid cells to ensure similar levels of the lava feature in the testing MDPs. Our development environment is 70% dirt and 30% grass. Our testing MDP is 5% lava, 66.5% dirt, and 28.5% grass.

In the proof-of-concept experiments, we selected the proxy reward function uniformly at random. For latent rewards, we picked a proxy reward function that evaluated to +1 for

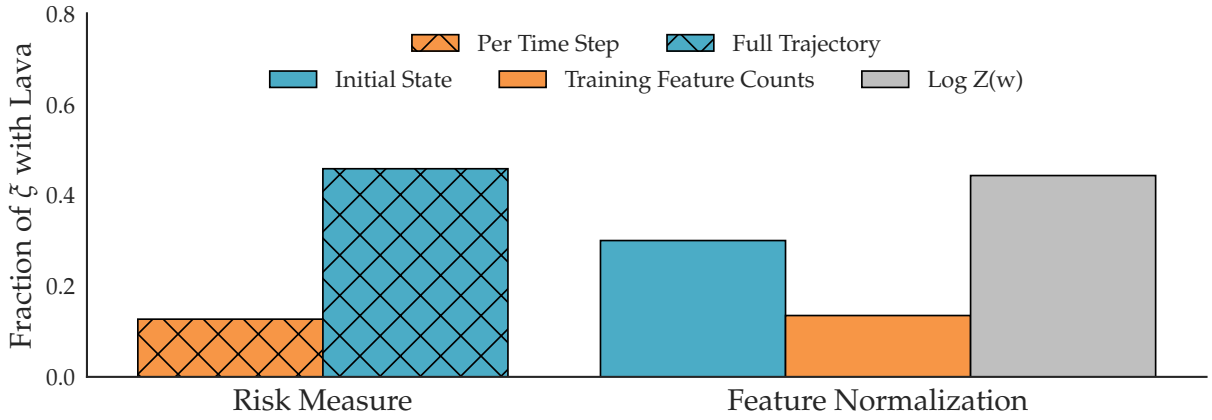


Figure 4.5: **Left:** We avoid side effects and reward hacking by computing a posterior distribution over reward function and then find a trajectory that performs well under the worst case reward function. This illustrates the impact of selecting this worst case independently per time step or once for the entire trajectory. Taking the minimum per time step increases robustness to the approximate inference algorithms used because we only need one particle in our sample posterior to capture the worst case for each grid cell type. For the full trajectory, we need a single particle to have inferred a worst case for *every* grid cell type at once. **Right:** The impact of changing the offsets c_i. "Initial State" fixes the value of the start state to be 0. "Training Feature Counts" sets an average feature value from the training MDP to be 0. "Log Z(w)" offsets each evaluation by the normalizing from the maximum entropy trajectory distribution. This means that the sum of rewards across a trajectory is the log probability of a trajectory.

target, $+.1$ for dirt, and $-.2$ for grass. To define a proxy on raw observations, we sampled 1000 examples of grass, dirt, and target and did a linear regression. With classifier features, we simply used the target rewards as the weights on the classified features. We used 50 dimensions for our feature vectors.

Independent Variables and Dependent Variabless. We measured the fraction of runs that encountered a lava cell on the test MDP as our dependent measure. This tells us the proportion of trajectories where the robot gets 'tricked' by the misspecified reward function; if a grid cell has never been seen then a conservative robot should plan to avoid it. We manipulate two factors: **literal-optimizer** and **Z-approx**. **literal-optimizer** is true if the robot interprets the proxy reward literally and false otherwise. **Z-approx** varies the approximation technique used to compute the IRD posterior. It varies across the two levels described above: sample to approximate the normalizing constant (**Sample-Z**) or use the normalizing constant from maximum entropy IRL (**MaxEnt-Z**) [181].

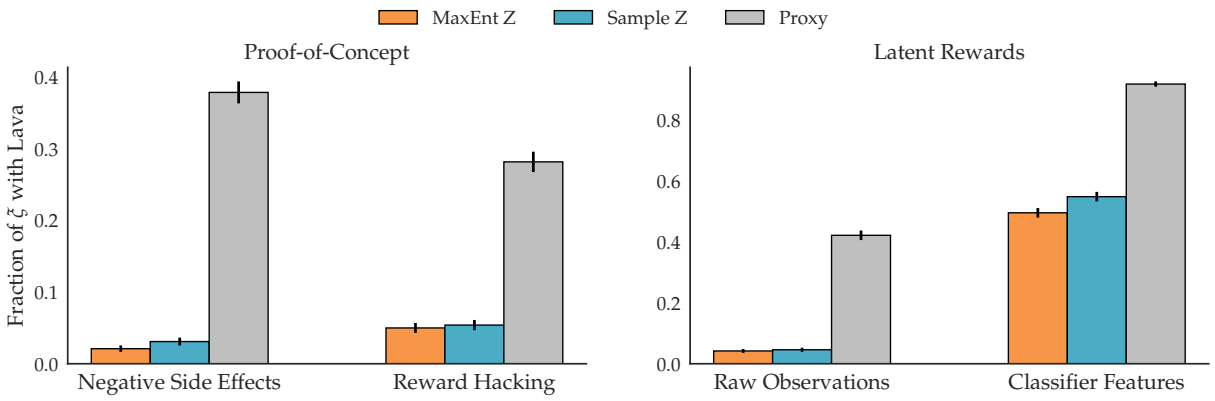


Figure 4.6: The results of our experiment comparing our proposed method to a baseline that directly plans with the proxy reward function. By solving an inverse reward design problem, we are able to create generic incentives to avoid unseen or novel states.

Comparing optimization methods

Evaluation. Before running the full experiment, we did an initial internal comparison to find the best-performing planning method. We did a full factorial across the factors with the side effect feature encoding and the reward hacking feature encoding. As a result, we had 6 conditions, arising from two possible settings of the **min-granularity** factor and the three options for the **reward-baseline** factor. Figure 4.5 shows the results.

We found that the biggest overall change came from the **min-granularity** factor. A bar plot is shown in Figure 4.5 (**left**). Independently minimizing per time step was substantially more robust. We hypothesize that this is a downstream effect of the approximate inference used. We sample from our belief to obtain a particle representation of the posterior. Independently minimizing means that we need a single particle to capture the worst case for each grid cell type. Performing this minimization across the full trajectory means that a single particle has to faithfully represent the worst case for *every* grid cell type.

We also saw substantial differences with respect to the **reward-baseline** factor. Figure 4.5 (**right**) shows a bar plot of the results. In this case, setting the common comparison point to be the average feature counts from the training MDP performed best. We believe this is because of the similarity between the train and test scenarios: although there is a new grid cell present, it is still usually possible to find a trajectory that is similar to those available in the training MDP. We hypothesize that correctly making this decision will depend on the situation.

Impact minimization in Lavaland

Results. Figure 4.6 compares the approaches. On the left, we see that IRD alleviates negative side effects (avoids the lava) and reward hacking (does not go as much on cells

that look deceptively like the target to one of the sensors). This is important, in that the same inference method generalizes across different consequences of misspecified rewards. Figure 4.3 shows example behaviors.

In the more realistic latent reward setting, the IRD agent avoids the lava cells despite the designer forgetting to penalize it, and despite not even having an indicator for it: because lava is latent in the space, and so reward functions that would *implicitly* penalize lava are as likely as the one actually specified, risk-averse planning avoids it.

We also see a distinction between raw observations and classifier features. The first essentially matches the proof-of-concept results (note the different axes scales), while the latter is much more difficult across all methods. The proxy performs worse because each grid cell is classified before being evaluated, so there is a relatively good chance that at least one of the lava cells is misclassified as target. IRD performs worse because the behaviors considered in inference plan in the already classified terrain: a non-linear transformation of the features. The inference must both determine a good linear reward function to match the behavior *and* discover the corresponding uncertainty about it. When the proxy is a linear function of raw observations, the first job is considerably easier.

4.3 Implementing a dynamic incentive protocol

In the previous section, we showed how to use IRD to create incentives for conservative optimization. In this section, we will propose a method that leverages reward inference to implement a dynamic reward design protocol, where **H** provides a sequence of proxy objectives to **R** in response to different context. Next, we use this protocol to explore what properties of an incentive design problem make it hard. Alternatively, we can think of this as identifying reward design protocols that minimize the cognitive load on **H**.

We start by showing how IRD can be extended to fuse proxy reward functions from multiple environments into a distribution over a single utility function that produces the same incentives in each development environment. This allows us to implement a dynamic reward specification protocol akin to the setup described in Section 3.3. Note, however, that this approach is different in that **R** optimizes for an estimate of utility that accounts for the entire sequence of proxy metrics.

Then, we run a sequence of human subject studies where participants use an interface to specify $\tilde{w}$ or a sequence of metrics $\tilde{w}^{(i)}$. Our first experiment compares two joint reward design protocols, one that optimizes the proxy directly and another that optimizes the mean of the IRD posterior, to this dynamic protocol. We follow up with another experiment that identifies properties of reward design problems that increase the cognitive load on participants.

Independent Reward Design

We suppose that there is a sequence of development environments $\tilde{E}^{\{1:N\}}$. The designer specifies N proxy reward functions $\tilde{w}^{\{1:N\}}$, one for each environment. Analogous to Section 4.2

we assume that $\widetilde{w}^{(n)}$ leads to successful task execution in environment $\tilde{E}^{(n)}$.

As before, we wish to determine a distribution over θ, represented as a set of weights w. In this case, we want to condition on the full sequence of observed proxies. This corresponds to

$$P\left(w^* \left| \widetilde{w}^{\{1:N\}}, \tilde{E}^{\{1:N\}} \right.\right) \propto \begin{array}{l} P\left(\widetilde{w}^{(N)} \left| \{w^*, E^{(N)}\}, \widetilde{w}^{\{1:N-1\}}, \tilde{E}^{\{1:N-1\}} \right.\right) \times \cdots \\ \times P\left(\widetilde{w}^{(1)} \left| w^*, \tilde{E}^{(1)} \right.\right) P_\theta\left(w^*\right) \end{array}. \tag{4.26}$$

To simplify the inference, we will suppose that the $\widetilde{w}^{(n)}$ are *conditionally independent*, given the true weights w^* and the development environments. We can therefore factor Equation 4.26 as

$$P\left(w = w^* \left| \widetilde{w}^{\{1:N\}}, \tilde{E}^{\{1:N\}} \right.\right) \quad \propto \quad P_\theta\left(w\right) \prod_{n=1}^{N} P\left(\widetilde{w}^{(n)} \left| w, \tilde{E}^{(n)} \right.\right). \tag{4.27}$$

This allows us to use the forward model from Section 4.2 directly in Equation 4.27. We use MCMC with the sampled normalizing constant to perform inference and let the agent model be the optimal policy $\pi^*_{\mathbf{R}}$, given the proxy. This allows us to use more efficient planning methods for the deterministic environments we consider.

Comparing joint and independent incentive specification

We begin by comparing independent reward design to two joint reward design baselines.
Nominal. The designer specifies a single proxy objective $\widetilde{w}$ that (ideally) induces the desired behavior jointly across all development environments $\tilde{E}^{(n)}$. The robot then optimizes proxy directly in the deployment environment.
Augmented. To remove confounds introduced by our inference method and isolate the changes caused by the interactive protocol, we also compare with a method that optimizes the mean of the IRD posterior for this joint proxy metric. In effective, this applies vanilla IRD to an MDP where $P_{\mathcal{S}}$ is a uniform distribution over the initial state of each $\tilde{E}^{(n)}$. Formally,

$$P\left(w = w^* \left| \widetilde{w}, \tilde{E}^{\{1:N\}} \right.\right) \quad \propto \quad P_\theta\left(w\right) \prod_{n} P\left(\widetilde{w} \left| w, \tilde{E}^{(n)} \right.\right) \tag{4.28}$$

where each term in the product is defined by Equation 4.15 for the appropriate development environment.

Independent Variables

We manipulated the reward design process with two conditions: independent and joint. In both cases, we presented the user with the same set of 5 environments, one of which is shown in Figure 4.7 (Right). For independent reward design, we presented the user with each

environment separately. For joint reward design, we presented the user with all environments at the same time.

In order to evaluate the performance of the algorithms, we needed access to the ground truth reward. Therefore, rather than asking users to design according to some internal preference, we showed them the trajectory that results from optimizing this ground truth reward. We then asked them to create a reward function that creates incentives for that behavior.

Dependent Variables

We looked at a combination of objective and subjective measurements of performance.
Objective Measures. We measured the time taken by the user to complete the task for each condition. We also measured the quality of the solution produced by reward design using regret computed on a set of 100 randomly-generated environments different from the training set.

Specifically, we employed 2 measures of performance using the output from reward design. The first (regret nominal) measures the regret when planning with the proxy for joint, and with the mean of the posterior over the true reward function for independent reward design.

To provide a fair comparison between the two conditions, we applied the machinery developed for generalizing designed rewards to joint reward design in Section 4.2. This approach, which we call augmented joint reward design, produces a distribution over the true reward function, which allows for more direct comparisons to independent reward design.

This leads to a second measure (regret IRD-augmented), whereby we compute the regret incurred when optimizing the mean of the distribution produced by augmented joint reward design.
Subjective Measures. We used the Likert scale questions in Table 4.3 to design a scale that captures the speed and ease of use for a reward design process. We first test the hypothesis that the interactive reward design protocol leads to less regret in the deployment environment.

Hypothesis 1. *The independent design process will lead to lower regret within a smaller amount of time, and with higher subjective evaluation of speed and ease of use.*

Lavaland with 5 terrain types

We begin by evaluating independent reward design in an adapted version of the Lavaland domain from Section 4.2. The two modifications are a diagonal action and an additional terrain type (leading to 5 total). We heuristically selected the sequence of environments to exhibit several characteristics of a typical, real-world robotics task.

1. When designing a reward function, a roboticist typically attempts to sample environments that capture a diverse set of conditions under which the robot operates. For example, when designing a reward for a self-driving car, it is wise to choose training

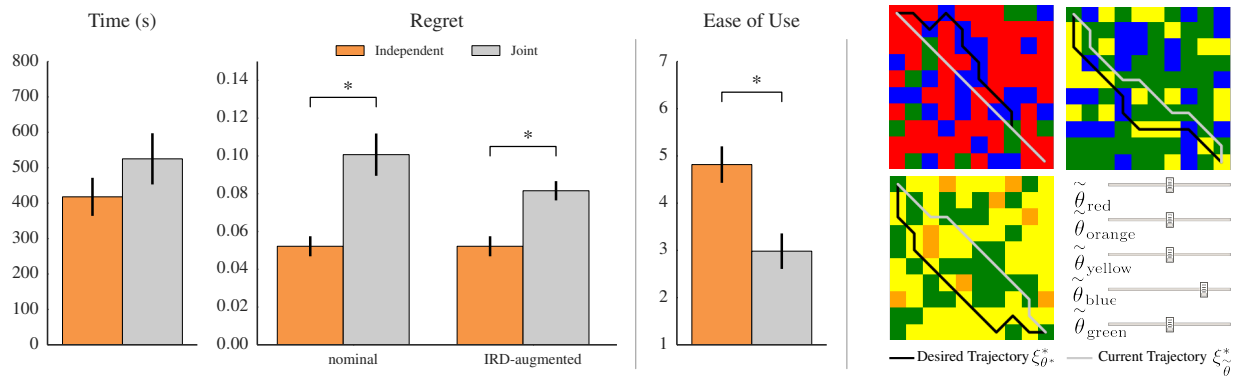


Figure 4.7: **Left:** The results of our study in grid worlds indicate that independent reward design is faster, leads to better performance, and is rated as easier to use. **Right:** Some of the grid world environments used in the study. Participants tuned the parameters of the reward function by using a set of sliders.

environments that exhibit a variety of weather, traffic conditions, road geometries, etc. Similarly, no two environments in our training set have the same distribution of features/colors.

2. We also capture the property that not all features will be present in a single environment. For example, we would rarely expect a self-driving car to experience all weather and traffic conditions, road geometries, etc. in a single training environment. Likewise, we chose the set such that no environment contains all of the features.

Participants and Allocation. We chose a within-subjects design to improve reliability, and controlled for the learning effect by putting independent always first (this may have slightly disadvantaged the independent condition). There were a total of 30 participants. All were from the United States and recruited through Amazon's Mechanical Turk (AMT) platform, with a minimum approval rating of 95% on AMT.

Results: Objective Measures. We first ran a repeated-measures ANOVA with reward design process as a factor and user ID as a random effect on the total time and the regret.

We found that independent outperformed joint across the board: the reward designed using the independent process had significantly lower regret ($F(1, 30) = 10.32$, $p < .01$), and took significantly lower user time to design ($F(1, 30) = 4.61$, $p = .04$).

Because IRD-based inference is also meant to generalize the designed rewards, we also tested the IRD-augmented joint reward. IRD did improve the regret of the jointly designed reward, as expected. The regret was still significantly lower for independent ($F(1, 30) = 15.33$, $p < .001$), however.

Figure 4.7 (**left**) plots the results. Supporting our hypothesis, we see that at least for this kind of problem the independent approach enables users to design *better* rewards *faster*.

Likert Questions
Q1: It was easy to complete [process].
Q2: I had a harder time with [process].
Q3: [process] was fast to complete.
Q4: [process] took fewer runs.
Q5: [process] was frustrating.
Q6: I went back and forth between different slider values a lot on [process].

Table 4.1: The set of questions used to define the subjective measures in our user study.

Results: Subjective Measures. The inter-item reliability of our scale was high (Cronbach's $\alpha = .97$). We thus averaged our items into one rating and ran a repeated measures ANOVA on that score. We found that independent led to significantly higher ratings than joint ($F(1, 30) = 33.63$, $p < .0001$). The mean rating went from 2.35 for joint to 5.49 for independent.

Remark 7. *Overall, the independent reward design protocol allowed users to design proxy metrics that generalize better than the joint proxy design baselines, in less time. Furthermore, this protocol was subjectively more preferred.*

Reward design for a 7-DoF arm

Next, we tested Hypothesis 1 in a robot motion planning problem for a Jaco 7-DoF arm planning trajectories in a household environment. We employ TrajOpt [134], an optimization-based motion planner, within a lightweight ROS/RViz visualization and simulation framework. Each environment contains a different configuration of a table, a vase, and a human. We used linear reward functions. Our features were radial basis function distances from the end-effector to: 1) the human's torso; 2)the human's head; 3)the vase; and 4) the distance from the end-effector to the table. These features are illustrated in Figure 4.8.

Participants specified the weights on the 4 features by tuning sliders, as in the grid worlds study. We asked participants to design a reward that induces a trajectory close to a ground truth trajectory that is visualized. This ground truth trajectory is the result of optimizing a ground truth reward function w^* that is hidden from the participants.

Participants and Allocation. We chose a within-subjects design to improve reliability, and counterbalancing to mitigate strong learning effects observed in the pilot. There were a total of 60 participants. All were from the United States and recruited through AMT with a minimum approval rating of 95%.

Results: Objective Measures. Our results in the manipulation domain are analogous to the grid world domain. We found a significant decrease in time ($F(1, 59) = 10.97$, $p < .01$) and regret (both nominal, $F(1, 59) = 38.66$, $p < .0001$, and augmented, $F(1, 59) = 54.68$, $p < .0001$) when users employed independent for designing rewards, as illustrated in Figure 4.9.

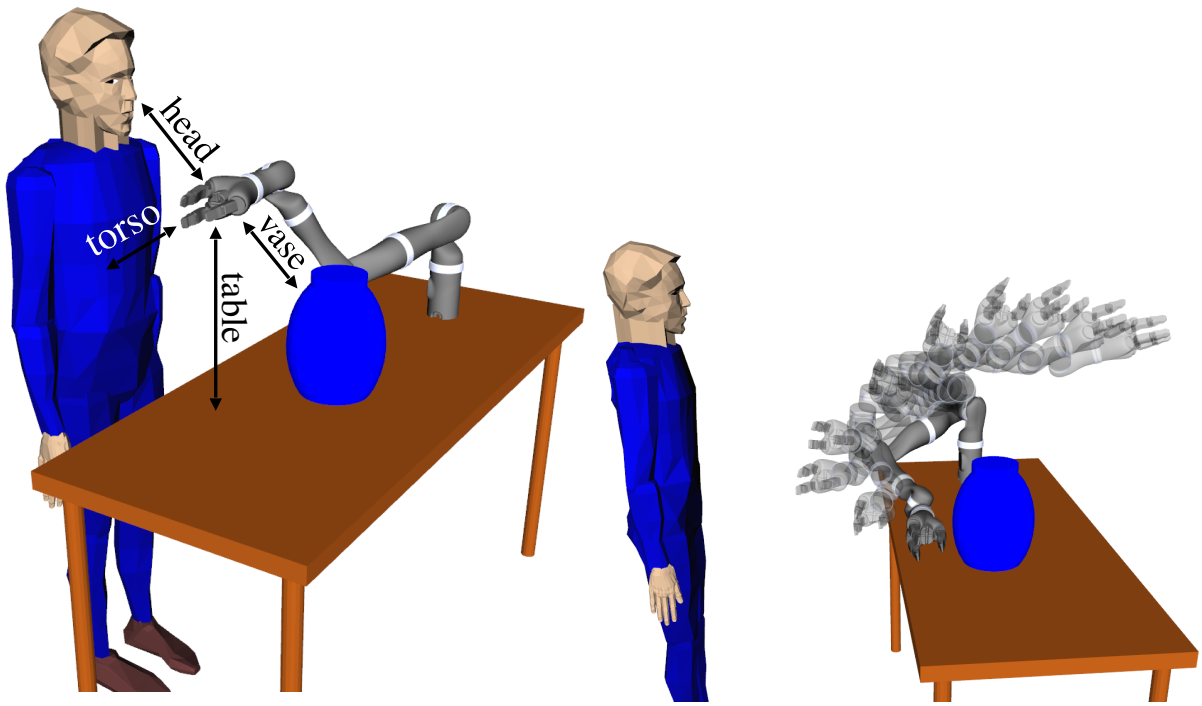


Figure 4.8: Two of the environments used in 7-DoF arm reward design study. Participants were asked to tune the weights on 4 features to achieve a desired trajectory: RBF distance to the human's head, torso, RBF distance to the vase, and distance from the table. After changing changing these weights, users were able to forward-simulate the trajectory to evaluate the induced behavior, as shown on the right.

Results: Subjective Measures. We found that users found independent to be significantly easier to use than joint in designing rewards ($F(1, 59) = 40.76$, $p < .0001$), as shown in Figure 4.9.

Overall, our results for manipulation follow those in the grid world domain: users perform better, faster, with the independent process in addition to subjectively preferring it to the joint process.

What makes reward design hard?

Next, we use controlled instances of reward design problems in the Lavaland domain to identify properties of development environments $\tilde{E}^{\{1:N\}}$ impact task completion time, performance, and subjective assessment.

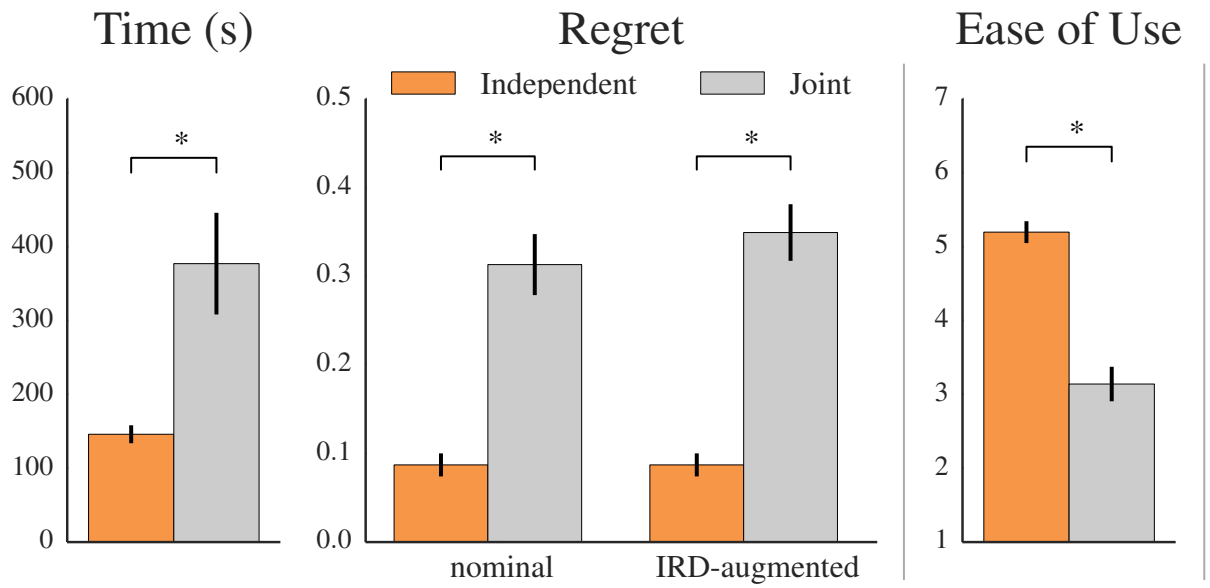


Figure 4.9: The results of our study on the Jaco 7-DOF arm are consistent with the previous study in grid worlds, further illustrating the potential benefits of independent reward design.

Changing the feasible reward set size

One way to characterize the difficulty of a reward design problem is by examining the *feasible rewards set* $\mathcal{F}_\epsilon(M, \tau)$: the set of proxy metrics that produce the desired trajectory τ in a given environment, within some tolerance $\epsilon \geq 0$. That is,

$$\mathcal{F}_\epsilon(E, \tau) = \{w \in \mathbb{R}^k : d(\tau, \tau_{w^*}) \leq \epsilon\}, \tag{4.29}$$

where $\tau_{w^*} = \text{argmax}_\tau \mathcal{U}(\tau; w^*)$ subject to the dynamics of environment E, $d : \mathcal{S}^* \times \mathcal{S}^* \to \mathbb{R}$ is a distance metric on trajectories, and $\epsilon \geq 0$ is a small constant. We vary both the size of $\mathcal{F}_\epsilon(E^{(n)}, \tau)$, as well as the intersection over all $\mathcal{F}_\epsilon(E^{(n)}, \tau)$, for $n = 1, \ldots, N$. For environments with discrete state and action spaces, we let $\epsilon = 0$, so that trajectories in the feasible set must match exactly the true trajectory.

We formulated 3 hypotheses for this experiment.

Hypothesis 2. *The size of the feasible set for an environment negatively affects the amount of time taken for that environment in the independent condition (i.e. larger feasible set leads to less time spent designing the reward).*

Hypothesis 3. *The size of the intersection of feasible sets negatively affects the amount of time taken in the joint condition (again, larger intersection set means smaller amount of time).*

We still believe that on average across this variety of environments, independent leads to better results and so we also test Hypothesis 1.

Independent Variables. We manipulate the reward design process (joint vs. independent), the average feasible set size over the set of environments (large vs. small), and the size of the intersection of the feasible sets for all environments (large vs. small).

We chose the categories of feasible sets using thresholds selected by examining the feasible set sizes and intersection sizes in the previous studies. To choose the environment sets for each of the 4 categories, we followed a simple procedure. We sampled sets of 5 environments, and computed the feasible set size and intersection size for each set. Then we chose 1 set from each category randomly to use in this study.

Dependent Measures. We use the same measures as in the previous experiment.

Participants and Allocation. Participants were recruited as in the grid worlds study, except now we recruited 20 participants per environment set for a total of 80.

Analysis. In this experiment, we attempt to characterize the difficulty of reward design on a set of environments by two properties: the feasible reward set size for each individual environment, and the intersection of feasible reward sets across the training set. Our results support the hypotheses that environments with *large* feasible reward sets are typically *easier*, and joint performs best for training sets with *large* intersections of feasible reward sets. Overall, however, independent still performs well in all cases.

To test Hypothesis 2, we fit the amount of time spent for each environment in the independent condition by the size of the feasible set for each user and environment, and found a significant negative effect as hypothesized ($F(1, 568) = 14.98$, $p < .0001$). This supports Hypothesis 2. Figure 4.10 (Bottom) shows a scatterplot of time by feasible set size.

To test Hypothesis 3, we fit the amount of time spent designing the joint reward by the size of the intersection of feasible sets. There was a marginal negative effect on time ($F(1, 113) = 2.55$, $p = .11$). This provides some partial support to Hypothesis 3.

To test Hypothesis 1, we ran a fully factorial repeated-measures ANOVA for time and regret using all 3 factors: design process, average size of feasible set (large vs. small), and size of intersection of feasible sets (large vs. small).

For time, we found two significant effects: process and intersections size. Independent indeed led to significantly less time taken *even across this wider set of environments* that is meant to balance the scales between independent and joint ($F(1, 163) = 10.27$, $p < .01$). And indeed, larger intersection size led to less time ($F(1, 163) = 8.93$, $p < .01$).

For regret, we again found a significant effect for process, with independent leading to significantly lower regret ($F(1, 163) = 19.66$, $p < .0001$). Smaller feasible sets and smaller intersections led to significantly lower regret too ($F(1, 163) = 38.69$, $p < .0001$ for feasible sets and $F(1, 163) = 8.1$, $p < .01$ for intersections). While smaller feasible reward sets are typically more challenging for the reward designer, they are in fact more informative – often smaller sets mean that when designers identify the desired reward, there are few others that would induce the same desired behavior. Informally, this means that this recovered reward is more likely to be close to the true reward, and hence will generalize well to new environments and induce lower regret. But there were also interaction effects: between process and feasible

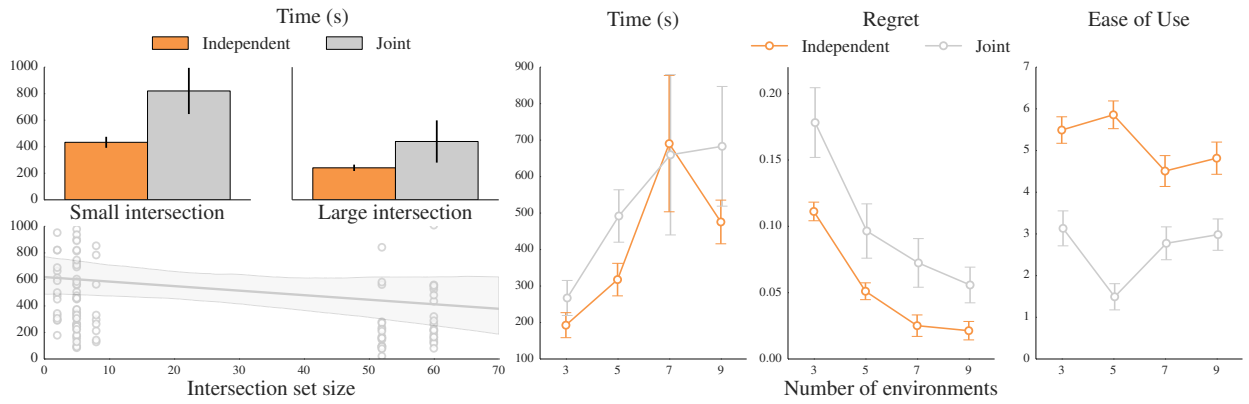


Figure 4.10: **Left:** Across small or large feasible sets and intersections sizes, independent still outperforms joint design. A smaller intersection size makes the problem harder, as expected. **Right:** As the number of environments increases, we still see benefits in regret and ease of use, but the time difference is less pronounced.

set size, and between feasible set size and intersection size. We did posthocs with Tukey HSD. We found that while independent improved regret significantly for large feasible sets (the setting where independent works best, because each environment is easy; $p < .0001$), the improvement was smaller and no longer significant for the small feasible sets. The interaction between the two size variables revealed that when both the sets and their intersection are large, the regret is the highest; when they are both small, the regret is lowest; when one of them is small and the other large, the regret lies somewhere in between the other two conditions.

The results for both time and regret support Hypothesis 1, and the results for time are summarized in Figure 4.10 (**left**). Overall, our result from the main study generalized to this wider range of environments: independent still lead to less time taken and lower regret. However, it does seem like environments with very small feasible sets would be difficult with the independent process, almost as difficult as with joint.

Remark 8. *Independent reward design protocols work best when each environment is simple, but getting a reward that works across* all *environments is hard.*

Changing the number of environments

Independent Variable. We manipulated the number of environments in the training set. Starting from the set of environments used in the primary study, we either randomly removed or added environments to achieve set sizes of 3, 7, and 9. When adding, we implemented a rejection sampling algorithm for heuristically avoiding environment sets that would be trivially-solved by a reward designer.

Hypothesis. Unlike before, we did not have a clear hypothesis prior to the study: while increasing the number of environments should make joint reward design more difficult, the effort required for independent reward design should scale linearly with the number of environments as well.
Participants and Allocation. We recruited participants using the strategy used in the primary study, except we recruited 20 users per condition, for a total of 80 users.
Analysis. We conducted a repeated measures factorial ANOVA with number of environments and process as factors, for each of our dependent measures. We found that the number of environments influenced many of the metrics. As shown in Figure 4.10 (Right), independent consistently outperformed joint in terms of regret ($F(1, 161) = 9.68$, $p < .01$) and ease of use ($F(1, 161) = 20.65$, $p < .0001$). We also saw that the time taken went up with the number of environments ($F(3, 159) = 4.94$, $p < .01$); this is unsurprising, as designing a reward for more environments should require more effort.

Furthermore, we observe that independent is most time efficient with respect to joint when there are a moderate number of environments (in this study, 5). If there are too few environments (in this study, 3) or too many environments (in this study, 7 or 9), then the differences in time taken are not significant. This follows our intuition about the regimes in which independent or joint reward design should be used.

This follows our intuition — if there are only a small number of environments to consider simultaneously, then joint should be an effective strategy for designing a reward; alternatively, if there are too many, then the advantages of divide-and-conquer diminish as the number of reward design "subproblems" to be solved grows.

Overall, we observe that while the reward recovered by independent induces lower regret and independent is consistently rated as easier to use, there is less of an advantage with respect to reward design time if the number of environments is *either* very small or very large. If it is too small, then there is little to be gained. If it is too large, the linear increase in costs eventually adds up.

Changing the fraction of features present

Our next experiment looked at the impact of the number of features per environment on overall performance. We vary the number of features per environment from 2 to 5 out of 5 total.
Independent Variables. We manipulated the fraction of the total features F present in each environment, varying from $F = 2/5$ to $F = 5/5$, with 5 possible relevant features in this domain (i.e. 5 different terrains in the grid world). Because we are limited to 5 features, we are actually studying the effect of varying the *fraction* of relevant features per environment on the performance of reward design.

In total, we had 5 sets of environments (4 on top of our set from the primary study). We also manipulated the reward design process as before.

Hypothesis 4. *Independent reward design outperforms joint when there are some, but not all of the total number of features possible per environment.*

We expect to see that independent has performance on par with joint when all features are present in each environment – then, the designer has to consider the effect of all features on the induced behavior, and hence must specify weights on *all* features for *all* training environments, the same as in the joint condition. We also expect the same performance between the two conditions when there are only 2 of 5 features present in each environment, because identifying the trade-offs between features becomes trivial.

Participants and Allocation. Participants were recruited as for the grid worlds study, except now we recruited 20 participants per environment set for a total of 80.

Results. Taken as a whole, the data aligns well with our hypothesis. Independent tends to outperform joint with a moderate fraction of the total number of features (3/5 or 4/5), but the two methods perform about the same when the minimum (2/5) or maximum (5/5) fraction of features are present. These results are summarized in Figure 4.11.

We ran a repeated measures factorial ANOVAs, with results largely supporting Hypothesis 4. The effect was most clear in the subjective measures, where we saw an interaction effect between the two factors on our ease of use scale– the difference between user preference on $F = 3/5$ and $F = 4/5$ was so strong, that even the Tukey HSD posthoc, which compensates for making 28 comparisons, found a significant improvement of independent over joint for those feature numbers ($p < .0001$ and $p < .02$). For other measures the difference on these two features was not strong enough to survive this level of compensation for multiple comparisons, but of course planned contrasts of independent vs. joint on each of the number of features support the improvement in both regret and time.

As a result, we find support for Hypothesis 4. Overall, independent outperforms joint when each environment only contains a subset of the total number of features relevant to the task.

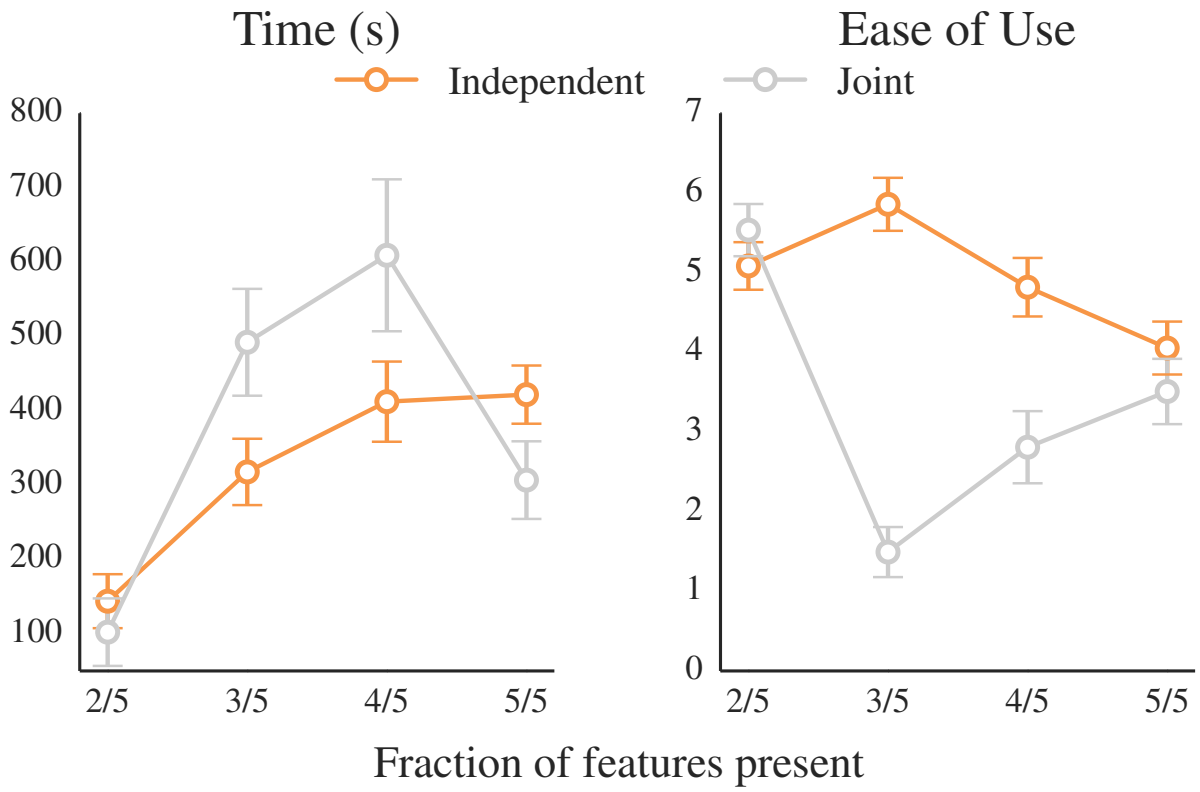


Figure 4.11: Independent reward design performs best in comparison with joint reward design when around half of the total number of relevant features are present in each environment. This is because each reward design problem is simple, but still provides a good amount of information about the overall objective.

Chapter 5

Pedagogy

So far we have argued two points. First, in Chapter 3 we argued that there is a principal—agent alignment problem caused by incomplete specification of incentives for AI systems. Furthermore, this alignment can be mitigated in theory by 1) regularizing optimization to reduce the strength of incentives and scope of changes and 2) using dynamic incentive schemes to adapt incentives to changes in the world.

Then, in Chapter 4 we argued that uncertainty about utility evaluations facilitates dynamic and pragmatic incentive design protocols. Section 4.1 shows how uncertainty about utility evaluations creates incentives to seek oversight and balance those incentives against the costs therein. Section 4.2 uses uncertainty to condition a distribution of objectives on an observed proxy and a development environment. We show how to use this to implement pragmatic optimization, that accounts for the context incentives were designed in. Then, in Section 4.3 we introduced a dynamic reward design protocol that reduces designer effort and reduces regret on held out problem instances.

We have, so far, modelled assistance problems as PoMDPs. That is, we have examined planning problems where observations of the principal's behavior provide information about the utility evaluation of states. In doing so, we model $\mathbf{H}$ as a Markov observation distribution: $\mathbf{H}$'s behavior depends on their policy $\pi_{\mathbf{H}}$, their type θ, and the state of the world s. While this is a useful model, it is wrong in potentially meaningful ways.

In this chapter, we will relax this constraint—motivated by three observations:

1. $\pi_{\mathbf{H}}$ is a highly influential component of the environment. Different choices for $\pi_{\mathbf{H}}$ can lead to large changes in the utility generated by $\mathbf{R} \circ \mathbf{H}$;

2. $\mathbf{H}$ may often be better understood as a strategic actor, *not* a static part of the environment. The primary consequence of this observation is that $\pi_{\mathbf{H}}$ can depend on $\pi_{\mathbf{R}}$ and may be *pedagogic*—providing extra information about θ in exchange for future utility generated by a better informed $\mathbf{R}$; and

3. It may be possible, through $\mathbf{R}$'s actions or interaction design, to influence $\mathbf{H}$'s policy to be more pedagogic with a more accurate world model, allowing $\mathbf{R} \circ \mathbf{H}$ to produce

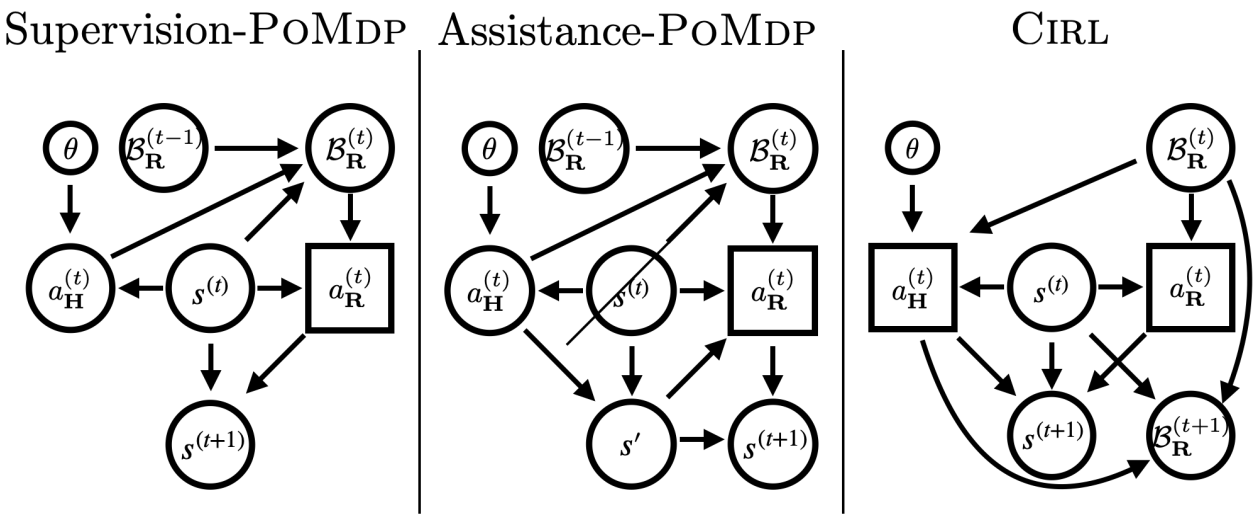


Figure 5.1: A comparison of the conditional dependencies for the sequential decision making models considered in this dissertation. **Left**: In a supervision-POMDP, **H** follows a state-based policy and produces observations, based on their type θ, **R**'s action space. **Middle**: In an assistance-POMDP, **H** and **R** have potentially distinct action spaces that both affect the world state. **Right:** In Section 5.2 we introduce *cooperative inverse reinforcement learning*, where **H** is modelled as a strategic actor who takes actions based on their type, the world state, *and* **R**'s current belief about θ. This represents **H**'s incentive to take information revealing actions that improve performance of the human–robot team, $\mathbf{R} \circ \mathbf{H}$.

more utility.

We begin in Section 5.1 by analyzing **H**'s incentives in assistance-POMDPs. First, we return to the supervision-POMDP from Section 3.1, where **H** directly labels current states with their preferred action and show how the value of β changes the information content of $\pi_{\mathbf{H}}$ and impacts the relative value of $\mathbf{R} \circ \mathbf{H}$. Then, we consider the optimal behavior of a demonstrator *that is aware of, and cares about,* **R***'s future behavior.* We show that pedagogic demonstrations can substantially increase the performance of a (learned) $\pi_{\mathbf{R}}$.

Then, in Section 5.2, we formalize *cooperative inverse reinforcement learning* (CIRL): a simple class of *assistance games*. CIRL generalizes assistance-POMDPs. It is a two-player game with jointly observed world state and asymmetrically observed static utility evaluations (i.e., θ is static and private to **H**). We show how to exponentially reduce the complexity of computing optimal strategy pairs, compared with general solvers of partial information games, and adapt approximate and exact POMDP planning algorithms to solve CIRL games. Figure 5.1 shows a comparison between the conditional dependencies for supervision-POMDPs, assistance-POMDPs, and CIRL.

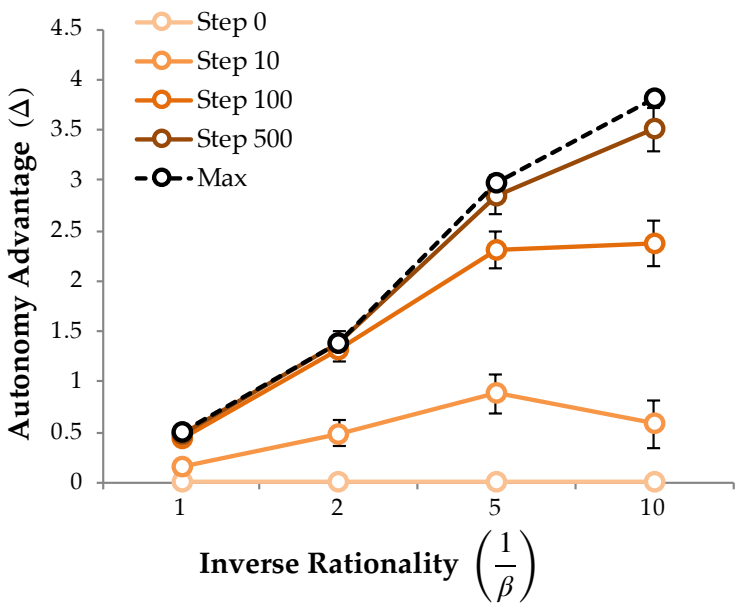


Figure 5.2: A plot of the autonomy advantage $\Delta^{(t)}_{\mathbf{R}\circ\mathbf{H}}$, the amount of additional utility $\mathbf{R}\circ\mathbf{H}$ produces at time t, compared with the teleoperation policy $\pi^{\text{TELEOP}}_{\mathbf{R}}$, against $\mathbf{H}$'s inverse rationality $\frac{1}{\beta}$. At the left side of the plot, $\mathbf{H}$ behaves nearly optimally, and so $\Delta_{\mathbf{R}\circ\mathbf{H}}$ is low. On the right, $\beta \to 0$, so $\mathbf{H}$ is more likely to take suboptimal actions. As a result $\Delta^{(\infty)}_{\mathbf{R}\circ\mathbf{H}}$ is large, indicating that $\mathbf{R}$ is very valuable in the limit. Note that for $t < \infty$ this relationship may be non-monotonic. β large has less room for improvement, but also reveals information about θ faster.

5.1 Pedagogic principals are easier to assist

We start by analyzing assistance-POMDPs with respect to changes in $\pi_{\mathbf{H}}$. We show that, if $\mathbf{H}$ is aware of the potential utility $\mathbf{R}$ can produce, then they have an incentive to be *pedagogic*: take actions that are suboptimal in isolation but increase $V^{R\text{policy}}$. Our first example considers $\mathbf{H}$'s level of noisy rationality in the supervision-POMDP from Section 3.1.

Choosing an optimal level of noisy rationality

Recall the supervision-POMDP from Definition 13, where $\mathbf{H}$ directly labels current states with their preferred action. The state is a matrix that encodes feature values for each of the K actions and new states are drawn independently from the initial distribution $P_{\mathcal{S}}$.

Figure 5.2 plots $\mathbf{H}$'s inverse rationality $\frac{1}{\beta}$ against $\Delta_{\mathbf{R}\circ\mathbf{H}}$, the additional value that $\mathbf{R}\circ\mathbf{H}$ generates compared to $\mathbf{H}$ acting alone. Recall that $\pi_{\mathbf{H}}$ is the noisily rational policy

$$\pi^{\beta}_{\mathbf{H}}(s;\theta) \propto \quad \exp\left(\beta Q_{\theta}(s,a)\right).$$

When $\pi_{\mathbf{H}}$ is highly irrational (i.e., $\beta \to 0$), it provides very little information per label, so $\mathbf{R}$ learns slowly, but $\mathbf{R} \circ \mathbf{H}$ is very valuable in the long run. When β is larger (i.e., $\frac{1}{\beta} \to 0$), $\pi_{\mathbf{H}}^{\beta}$ reveals more information about θ, but $\mathbf{R} \circ \mathbf{H}$ relies on $\mathbf{H}$ to do all the work. This is an issue as increasing β typically requires, e.g., increased attention and effort from $\mathbf{H}$.

Generating pedagogic demonstrations

The previous section showed that $\mathbf{R}$'s behavior creates incentives for $\mathbf{H}$ to be more informative about θ. However, we only allowed $\mathbf{H}$ to choose β. In effect, this limits the strategic component of $\mathbf{H}$'s behavior to the level of noise in their actions. In general, however, $\mathbf{H}$ may optimize any aspect of $\pi_{\mathbf{H}}$ to coordinate with $\mathbf{R}$. $\mathbf{R}$ also optimizes $\pi_{\mathbf{R}}$ in response. As a result, the optimal configuration of $\mathbf{R} \circ \mathbf{H}$ is in *pedagogic equilibrium*, where $\pi_{\mathbf{R}}$ and $\pi_{\mathbf{H}}$ are mutual best responses. We look at the general problem of computing a pedagogic equilibrium efficiently in Section 5.2. For now, we characterize pedagogic behavior in *apprenticeship games*: assistance games with two phases, a *development* phase where $\mathbf{H}$ takes actions and a *deployment* phase where $\mathbf{R}$ takes actions. This is analogous to the development and deployment environments from Chapter 4, with the change that $\mathbf{H}$ may take actions other than specifying a proxy metric.

We start by characterizing pedagogic equilibrium in an apprenticeship game with 3 actions each for $\mathbf{H}$ and $\mathbf{R}$.

Example. Consider an apprenticeship game where $\mathbf{R}$ needs to help $\mathbf{H}$ make office supplies. $\mathbf{H}$ and $\mathbf{R}$ can make paperclips and staples and θ describes $\mathbf{H}$'s preference for paperclips vs staples. We model the problem as an apprenticeship game, where the learning and deployment phase each consist of an individual action.

The world state in this problem is a tuple (p_s, q_s, t) where p_s and q_s respectively represent the number of paperclips and staples $\mathbf{H}$ owns. t is the round number. An action is a tuple (p_a, q_a) that produces p_a paperclips and q_a staples. The human can make 2 items total: $\mathcal{A}_{\mathbf{H}} = \{(0,2),(1,1),(2,0)\}$. The robot has different capabilities. It can make 50 units of each item or it can choose to make 90 of a single item: $\mathcal{A}_{\mathbf{R}} = \{(0,90),(50,50),(90,0)\}$.

We let $\Theta = [0,1]$ and define $\mathcal{U}$ so that w indicates the relative preference between paperclips and staples: $\mathcal{U}(s,(p_a,q_a);w) = wp_a + (1-w)q_a$. $\mathbf{R}$'s action is ignored when $t = 0$ and $\mathbf{H}$'s is ignored when $t = 1$. At $t = 2$, the game is over, so we transition to a sink state, $(0,0,2)$. Initially, there are no paperclips or staples and P_θ is a uniform distribution on $[0,1]$.

$\mathbf{H}$ only acts in the initial state, so $\pi_{\mathbf{H}}$ can be entirely described by a single decision rule $v_{\mathbf{H}} : [0,1] \to \mathcal{A}_{\mathbf{H}}$. $\mathbf{R}$ only observes one action from $\mathbf{H}$ and so $\mathbf{R}$'s policy is a mapping from $a_{\mathbf{H}}$ to $a_{\mathbf{R}}$, $\pi_{\mathbf{R}} : \mathcal{A}_{\mathbf{H}} \to \mathcal{A}_{\mathbf{R}}$.

It is simplest to analyze the deployment phase first. $\mathbf{R}$ is the only actor in this phase so it get no more observations of its reward. Thus, we can leverage the result from Ramachadran and Amir [122] to show that $\mathbf{R}$'s optimal deployment policy in an apprenticeship game maximizes utility for the mean value of θ.

Corollary 13.1. *Let M be an apprenticeship game. In the deployment phase, the optimal policy for $\mathbf{R}$ maximizes reward in the* MDP *induced by the mean w from $\mathbf{R}$'s belief.*

Proof. In the deployment phase, $\mathbf{R}$ optimizes $\pi_{\mathbf{R}}$ for a static distribution of θ. This meets the conditions of Theorem 3 from Ramachadran and Amir [122] and the result follows directly. □

In our example, suppose that $\pi_{\mathbf{H}}$ selects $(0, 2)$ if $w \in [0, \frac{1}{3})$, $(1, 1)$ if $w \in [\frac{1}{3}, \frac{2}{3}]$ and $(2, 0)$ otherwise. $\mathbf{R}$ begins with a uniform prior on w so observing, e.g., $a_{\mathbf{H}} = (0, 2)$ leads to a posterior distribution that is uniform on $[0, \frac{1}{3})$. Corollary 13.1 shows that the optimal action maximizes reward for the mean w so an optimal $\mathbf{R}$ behaves as though $w = \frac{1}{6}$ during the deployment phase.

Corollary 13.1 enables us to characterize the best response for $\mathbf{R}$ when myopically maximizes immediate reward. $\mathbf{R}$ should use Bayesian-IRL to compute the posterior over w during the learning phase and then act to maximize utility under the mean w in the deployment phase. Next, we will look at the incentives this induces for $\mathbf{H}$. We will show that $\mathbf{H}$'s best response is distinct from $\pi^{\beta}_{\mathbf{H}}$. In turn, this shows that, in general, the pair $\left(\pi^{\beta}_{\mathbf{H}}, \mathbf{br}\left(\pi^{\beta}_{\mathbf{H}}\right)\right)$ is *not* in pedagogic equilibrium.

Theorem 14. *There exist apprenticeship games where the best-response for $\mathbf{H}$ to an* IRL-$\mathbf{R}$ *deviates from the behavioral model used therein. In other words, there exist apprenticeship games such that*

$$\boldsymbol{br}\left(\boldsymbol{br}\left(\pi^{\beta}_{\mathbf{H}}\right)\right) \neq \pi^{\beta}_{\mathbf{H}}. \tag{5.1}$$

Proof. Our office supply example gives a counter example that shows the theorem.

An IRL-$\mathbf{R}$ assumes that $\mathbf{H}$ maximizes utility. Let $v_{\mathbf{E}}$ represent the decision rule for this policy. We can characterize $v_{\mathbf{E}}$ and its best response $\mathbf{br}(v_{\mathbf{E}})$ as follows:

$$v_{\mathbf{E}}(w) = \begin{cases} (0,2) & w < 0.5 \\ (1,1) & w = 0.5 \\ (2,0) & w > 0.5 \end{cases}, \tag{5.2}$$

$$\mathbf{br}\left(v_{\mathbf{E}}\right)\left(a_{\mathbf{H}}\right) = \begin{cases} (0,90) & a_{\mathbf{H}} = (0,2) \\ (50,50) & a_{\mathbf{H}} = (1,1) \\ (90,0) & a_{\mathbf{H}} = (2,0) \end{cases}. \tag{5.3}$$

Note that when $w = 0.49$ $\mathbf{H}$ would prefer $\mathbf{R}$ to choose (50, 50). $\mathbf{H}$ is willing to forgo immediate reward during the demonstration to communicate this to $\mathbf{R}$: the best response chooses $(1, 1)$ when $w = 0.49$.

More generally, when $\mathbf{H}$ accounts for $\mathbf{R}$'s actions under $\mathbf{br}\left(v_{\mathbf{E}}\right)$, $\mathbf{H}$ is faced with a choice between 0 paperclips and 92 staples, 51 of each, or 92 paperclips and 0 staples. It is straightforward to show that the optimal *pedagogic* decision rule is given by

$$v_{\mathbf{H}}(w) = \begin{cases} (0,2) & w < \frac{41}{92} \\ (1,1) & \frac{41}{92} \leq w \leq \frac{51}{92} \\ (2,0) & w > \frac{51}{92} \end{cases} . \tag{5.4}$$

This is distinct from Equation 5.2 so we conclude the result. □

Remark 9. *We should expect experienced users of apprenticeship learning systems to present demonstrations optimized for fast learning rather than demonstrations that maximize reward.*

Pedagogic demonstrations for inverse reinforcement learning

Now, we consider the problem of computing **H**'s best response when **R** is an IRL-**R** more generally. In the office supply problem, it is feasible to exhaustively search the space of joint policies. For non-trivial apprenticeship games, we need a more efficient approach. In this section, we present one such method: a heuristic approach that leverages a property of IRL with linear utility functions. We show that it is able to improve the utility produced by $\mathbf{R} \circ \mathbf{H}$, compared to expert demonstrations.

Specifically, we use a linear utility model so that $\mathcal{U}(s, a_{\mathbf{H}}, a_{\mathbf{R}}; w) = w^{\top}\phi(s)$. Standard results from the IRL literature show that policies with the same expected feature counts have the same value under *any* setting of θ [2]. Combined with Corollary 13.1, this implies that an optimal IRL-**R** will compute a policy that matches the observed feature counts from the development phase.

This suggests a simple approximation scheme. To compute a demonstration trajectory $\tau_{\mathbf{H}}$, first compute the feature counts **R** would observe in expectation from the true w and then select actions that maximize similarity to these target features. If ϕ_w are the expected feature counts induced by w then this scheme amounts to the following decision rule:

$$\tau_{\mathbf{H}} \leftarrow \arg\max_{\tau} \phi(\tau)^{\top} w - \eta ||\phi_w - \phi(\tau)||^2. \tag{5.5}$$

This rule selects a trajectory that trades off between the sum of rewards $\phi(\tau)^{\top} w$ and the feature dissimilarity $||\phi_w - \phi(\tau)||^2$. Note that this is generally distinct from the action selected by the demonstration-by-expert policy. The goal is to match the expected sum of features under a *distribution* of trajectories with the sum of features from a *single* trajectory. The correct measure of feature similarity is the regret a set of features counts induces relative to the utility **R** would collect if it knew the true w. Computing this similarity is expensive, so we use an ℓ_2 norm as a proxy measure of similarity.

Figure 5.3 shows an example comparison between demonstration-by-expert and the approximate best response policy in Section 5.1. The leftmost image is the ground truth reward function. Next to it are demonstration trajectories produced by these two policies. Each path is superimposed on the maximum a-posteriori reward function the robot infers from the demonstration. We can see that the demonstration-by-expert policy immediately goes to the highest reward and stays there. In contrast, the best-response policy moves to both areas

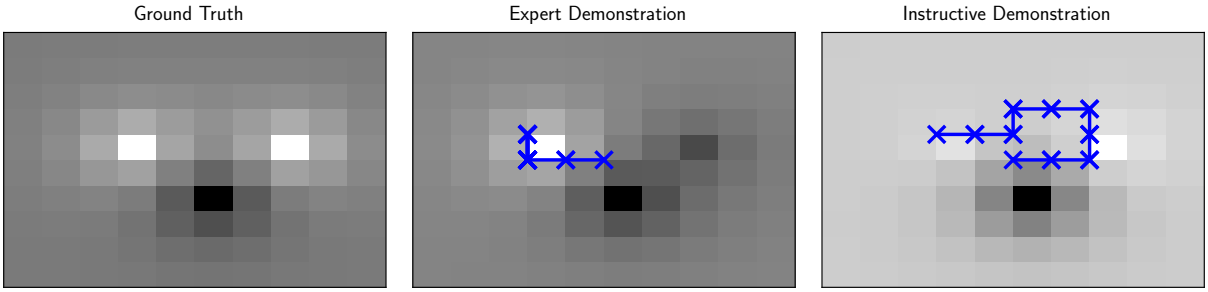


Figure 5.3: The difference between demonstration-by-expert and instructive demonstration in a 2D gridworld. The three features are radial basis functions from one of 3 locations in the center of the map. Left: The ground truth reward function. Lighter grid cells indicates areas of higher reward. Middle: The demonstration trajectory generated by the expert policy, superimposed on the maximum a-posteriori reward function the robot infers. The robot successfully learns where the maximum reward is, but little else. Right: An instructive demonstration generated by the algorithm in Section 5.1 superimposed on the maximum a-posteriori reward function that the robot infers. This demonstration highlights both points of high reward and so the robot learns a better estimate of the reward.

of high reward. The reward function the robot infers from the best response demonstration is much more representative of the true reward function, when compared with the reward function it infers from demonstration-by-expert.

Pedagogy improves learning

We tested this in a 2D navigation problem on a discrete grid. In the development phase of the game, **H** teleoperates a trajectory while **R** observes. In the deployment phase, **R** is placed in a random state and given control of the robot. We use a finite horizon H, and let the first $\frac{H}{2}$ timesteps be the development phase. There are N_ϕ state features defined as radial basis functions where the centers are common knowledge. Rewards are linear in these features and w. The initial world state is in the middle of the map. We use a uniform distribution on $[-1, 1]^{N_\phi}$ for the prior on w. Actions move in one of the four cardinal directions $\{N, S, E, W\}$.

Hypothesis 5. *When* **R** *is an* IRL-**R** *that matches features, the approximate best response policy from Section 5.1 cause* **R** *to produce more utility than demonstration-by-expert.*

Manipulated Variables. Our experiment consists of 2 factors: ***H-policy*** and ***num-features***. We make the assumption that **R** uses an IRL algorithm to compute its estimate of w during learning and maximizes reward under this estimate during deployment. We use Maximum-Entropy IRL [181] to implement **R**'s policy. ***H-policy*** varies **H**'s strategy $\pi_{\mathbf{H}}$ and has two levels: demonstration-by-expert ($\pi_{\mathbf{E}}$) and best-responder (**br**). In the $\pi_{\mathbf{E}}$ level

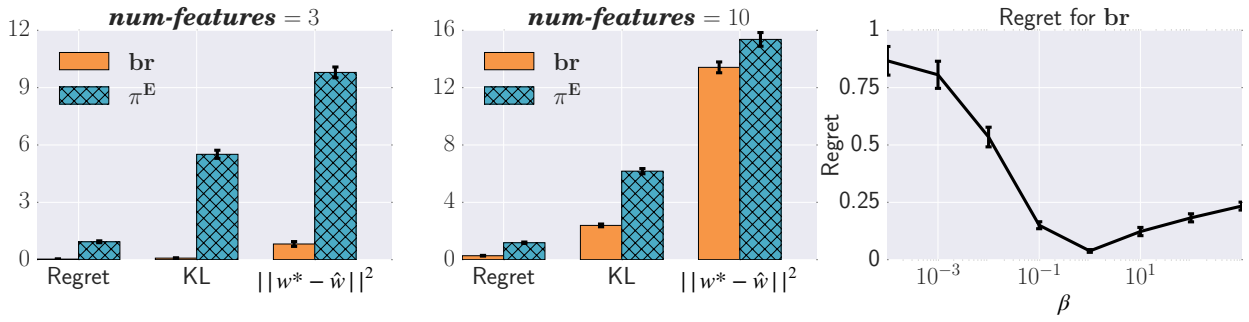


Figure 5.4: Left, Middle: Comparison of demonstration-by-expert ($\pi^{\mathbf{E}}$) with pedagogic demonstration (**br**). Lower numbers are better. Using the best response causes **R** to infer a better distribution over w so it does a better job of maximizing reward. Right: The regret of the instructive demonstration policy as a function of how optimal **R** expects **H** to be. $\beta = 0$ corresponds to a robot that expects purely random behavior and $\beta = \infty$ corresponds to a robot that expects optimal behavior. Regret is minimized for an intermediate value of β: if β is too small, then **R** learns nothing from its observations; if β is too large, then **R** expects many values of w to lead to the same trajectory so **H** has no way to differentiate those reward functions.

H maximizes reward during the demonstration. In the **br** level **H** uses the approximate pedagogical algorithm defined above to compute an approximate best response to $\pi_{\mathbf{R}}$. The trade-off between reward and communication η is set by cross-validation before the game begins. The ***num-features*** factor varies the dimensionality of ϕ across two levels: 3 features and 10 features. We do this to test whether and how the difference between experts and best-responders is affected by dimensionality. We use a factorial design that leads to 4 distinct conditions. We test each condition against a random sample of $N = 500$ different reward parameters. We use a within-subjects design with respect to the the **H-policy** factor so the same reward parameters are tested for $\pi_{\mathbf{E}}$ and **br**.

Dependent Measures. We use the regret with respect to a fully-observed setting where the robot knows the ground truth w as a measure of performance. We let $\hat{w}$ be the robot's estimate of the reward parameters and let w^* be the ground truth reward parameters. The primary measure is the *regret* of **R**'s policy: the difference between the value of the policy that maximizes the inferred reward $\hat{w}$ and the value of the policy that maximizes the true reward w^*. We also use two secondary measures. The first is the KL-divergence between the maximum-entropy trajectory distribution induced by $\hat{w}$ and the maximum-entropy trajectory distribution induced by w. Finally, we use the ℓ_2-norm between the vector or rewards defined by $\hat{w}$ and the vector induced by w^*.

Results. There was relatively little correlation between the measures (Cronbach's α of .47), so we ran a factorial repeated measures ANOVA for each measure. Across all measures, we found a significant effect for **H-*policy***, with **br** outperforming $\pi_{\mathbf{E}}$ on all measures as we

H-*policy*	***num-features*=3**			***num-features*=10**		
	Regret	KL-divergence	$\|\|w^* - \hat{w}\|\|^2$	Regret	KL-divergence	$\|\|w^* - \hat{w}\|\|^2$
$\pi_{\mathbf{E}}$	11.3	5.53	9.7	16.1	6.2	15.3
br	0.2	0.1	1.0	4.3	2.4	13.3

Table 5.1: Comparison of demonstration-by-expert ($\pi_{\mathbf{E}}$) with pedagogic demonstration (**br**). Lower numbers are better. Using the best response causes **R** to infer a better distribution over w so it does a better job of maximizing utility.

hypothesized (all with $F > 962$, $p < .0001$). We did find an interaction effect with ***num-features*** for KL-divergence and the ℓ_2-norm of the reward vector but post-hoc Tukey HSD showed **br** to always outperform $\pi_{\mathbf{E}}$. The interaction effect arises because the gap between the two levels of **H-*policy*** is larger with fewer reward parameters; we interpret this as evidence that ***num-features*** $= 3$ is an easier teaching problem for **H**. Figure 4.6 (Left, Middle) shows the dependent measures from our experiment.

5.2 Cooperative Inverse Reinforcement Learning

Figure 5.4 shows that **R**'s performance can be substantially improved by adopting a pedagogic demonstration strategy that reasons directly about the feature counts created for **R**. In this section, we define *cooperative inverse reinforcement learning* (IRL) a generalization of assistance-POMDPs where **H** is a strategic actor. As a result, **H**'s actions can also depend on the history of the interaction. We start with a formal definition of CIRL. Then, we define *pedagogic equilibrium* as a solution concept for CIRL games. We show how to efficiently compute pedagogic equilibrium in two steps: 1) we show that **R**'s belief $\mathcal{B}_{\mathbf{R}}$ is a *sufficient statistic* for utility maximizing strategies by reducing the problem to a POMDP with an exponentially sized action space; and 2) we leverage the information structure of that POMDP to speed up planning and show that CIRL games can be solved in roughly the amount of time it takes to solve a similarly sized POMDP.

A Formal Model of Principal–Agent Alignment

Definition 22. (Cooperative Inverse Reinforcement Learning)
*A cooperative inverse reinforcement learning (*CIRL*) game is defined as a tuple*

$$G = \langle \{\mathcal{S}, P_{\mathcal{S}}\}, \{\mathcal{A}_{\mathbf{R}}, \mathcal{A}_{\mathbf{H}}\}, T, \{\Theta, P_\theta\}, U \rangle : \tag{5.6}$$

$\mathcal{S}$ *A set of world states,* $s \in \mathcal{S}$;

$P_{\mathcal{S}}$ *A distribution over the initial state of the world.* $P_{\mathcal{S}} \in \Delta(\mathcal{S})$;

$\mathcal{A}_{\mathbf{H}}$ *A set of actions for* $\mathbf{H}$*,* $a_{\mathbf{H}} \in \mathcal{A}_{\mathbf{H}}$;

$\mathcal{A}_{\mathbf{R}}$ *A set of actions for* $\mathbf{R}$*,* $a_{\mathbf{R}} \in \mathcal{A}_{\mathbf{R}}$;

T *A transition distribution that determines the distribution over next states, given previous state and actions for* $\mathbf{H}$ *and* $\mathbf{R}$*.* $T : \mathcal{S} \times \mathcal{A}_{\mathbf{H}} \times \mathcal{A}_{\mathbf{R}} \to \Delta(\mathcal{S})$.

Θ *A space of* types *that determine* $\mathbf{H}$*'s utility function,* $\theta \in \Theta$;

P_θ *A distribution over* $\mathbf{H}$*'s type,* $P_\theta \in \Delta(\Theta)$;

$\mathcal{U}$ *A parameterized utility function that maps a state and type to a real number that represents* $\mathbf{H}$*'s utility,* $\mathcal{U}(s, a_{\mathbf{H}}, a_{\mathbf{R}}; \theta) : \mathcal{S} \times \mathcal{A}_{\mathbf{H}} \times \mathcal{A}_{\mathbf{R}}\Theta \to \mathbb{R}$;

γ *A discount factor that trades off between current and future utility.* $\gamma \in [0, 1)$.

In CIRL, we consider $\mathbf{H}$'s ability to anticipate $\mathbf{R}$'s actions and respond accordingly. This creates incentives to teach and leads to solutions where $\mathbf{H}$'s actions reveal more information about their type. In general, this increases the value $\mathbf{R}$ is able to provide. In practice, this changes the domain for $\mathbf{H}$'s policy to depend on their full information set. Formally,

$$\pi_{\mathbf{H}} \in \Theta \times (\mathcal{S} \times \mathcal{A}_{\mathbf{H}} \times \mathcal{A}_{\mathbf{R}})^* ; \pi_{\mathbf{R}} \in (\mathcal{S} \times \mathcal{A}_{\mathbf{H}} \times \mathcal{A}_{\mathbf{R}})^* . \tag{5.7}$$

We illustrate the dependencies in Figure 5.5 with $\mathcal{B}_{\mathbf{R}}$ standing in for the history of actions from both players.[1] Because $\pi_{\mathbf{R}}$ and $\pi_{\mathbf{H}}$ depend on the previous actions of the other player, this allows for communication and coordination. This interdependence also creates nuance in the choice of optimality criteria. This is because a solution to a CIRL game is a pair of strategies, $(\pi_{\mathbf{H}}, \pi^*_{\mathbf{R}})$, one for each actor. While it is straightforward (although often highly non-trivial) to imagine $\mathbf{R}$ implementing a given policy, people are a different story. We have lots of evidence that game theory does not predict human behavior perfectly.

To manage some of this complexity, we consider two related solution concepts for a CIRL game: one that defines optimality for $\mathbf{R}$ in isolation and one that describes optimal team performance. We will consider solutions that are robust to difference choices of $\pi_{\mathbf{H}}$ in Chapter 6. The optimality condition in isolation is the best-response criterion considered in the previous section. In a CIRL game it is defined as follows:

Definition 23. (Best Response)
Let $G = \langle \{\mathcal{S}, P_{\mathcal{S}}\}, \{\mathcal{A}_{\mathbf{R}}, \mathcal{A}_{\mathbf{H}}\}, T, \{\Theta, P_\theta\}, U \rangle$ *be a* CIRL *game. For a pair of strategies* $(\pi_{\mathbf{H}}, \pi_{\mathbf{R}})$*,* $\mathbf{R}$*'s policy* $\pi_{\mathbf{R}}$ *is a best response to* $\pi_{\mathbf{H}}$*, written* $\pi_{\mathbf{R}} \in \boldsymbol{br}(\pi_{\mathbf{H}})$*, iff*

$$\pi_{\mathbf{R}} \in \arg\max_{\pi} \ \mathbb{E}\left[\gamma^t \sum_t \mathcal{U}_\theta\left(s^{(t)}, a_{\mathbf{H}}^{(t)}, a_{\mathbf{R}}^{(t)}\right) \middle| \begin{array}{r} s^{(t+1)} \sim T\left(s^{(t)}, a_{\mathbf{H}}^{(t)}, a_{\mathbf{R}}^{(t)}\right); \\ a_{\mathbf{R}}^{(t)} \sim \pi_{\mathbf{R}}\left(s^{(t)}, \tau^{(t-1)}\right); \\ a_{\mathbf{H}}^{(t)} \sim \pi_{\mathbf{H}}\left(s^{(t)}, \tau^{(t-1)}; \theta\right) \end{array} \right]. \tag{5.8}$$

[1] We will show later in Theorem 17 that this restriction preserves the optimal strategy pair.

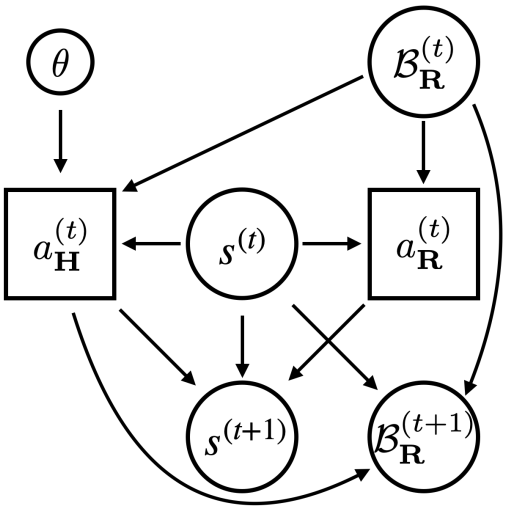


Figure 5.5: The conditional dependencies for a single timestep of a CIRL game. **H**'s action $a_{\mathbf{H}}^{(t)}$ depends on the world state $s^{(t)}$, their type θ and **R**'s belief state $\mathcal{B}_{\mathbf{R}}^{(t)}$. **R**'s action $a_{\mathbf{R}}^{(t)}$, on the other hand, only depends on the world state $s^{(t)}$ and its belief $\mathcal{B}_{\mathbf{R}}^{(t)}$. This depiction relies on Theorem 17 as, in general, game strategies can depend on the full state-action history.

Definition 23 describes a family of optimal policy for **R** as a function of **H**'s policy $\pi_{\mathbf{H}}$. The key difference between Equation 5.8 and the optimality criterion in assistance-POMDPs is that $\pi_{\mathbf{H}}$ is no longer a Markov component of the environment. This is no longer a POMDP because the (joint) state-action history τ changes the distribution of $a_{\mathbf{H}}$. As a result, **R** has to reason about the impact of its actions on **H**'s future behavior through mechanisms *other* than the world state. For example, action selection in the pedagogic demonstration policies from Section 5.1 depends on τ through the features observed thus far and the target feature counts. This leverages the fact that feature counts are a *sufficient statistic* for $\mathcal{B}_{\mathbf{R}}$ when the utility model is linear and **R** is an IRL-**R**. (In the next section, we show that optimal strategy pairs can always use **R**'s belief $\mathcal{B}_{\mathbf{R}}$ as a sufficient statistic for the interaction history.)

We say that a pair of strategies $(\pi_{\mathbf{H}}, \pi_{\mathbf{R}})$ are in *pedagogic equilibrium* when $(\pi_{\mathbf{H}}, \pi_{\mathbf{R}})$ are mutual best responses.

Definition 24. (Pedagogic Equilibrium)
Let $G = \langle \{\mathcal{S}, P_{\mathcal{S}}\}, \{\mathcal{A}_{\mathbf{R}}, \mathcal{A}_{\mathbf{H}}\}, T, \{\Theta, P_\theta\}, U \rangle$ be a CIRL *game. A strategy pair $(\pi_{\mathbf{H}}^*, \pi_{\mathbf{R}}^*)$ is in pedagogic equilibrium iff*

$$\pi_{\mathbf{H}}^* \in \arg\max_{\pi_{\mathbf{H}}} \mathbb{E}\left[\sum_t \gamma^t \mathcal{U}_\theta\left(s^{(t)}, a_{\mathbf{H}}^{(t)}, a_{\mathbf{R}}^{(t)}\right) \;\middle|\; \begin{array}{c} s^{(t+1)} \sim T\left(s^{(t)}, a_{\mathbf{H}}^{(t)}, a_{\mathbf{R}}^{(t)}\right); \\ a_{\mathbf{H}}^{(t)} \sim \pi_{\mathbf{H}}\left(s^{(t)}, \tau^{(t-1)}; \theta\right); \\ a_{\mathbf{R}}^{(t)} \sim \pi_{\mathbf{R}}^*\left(s^{(t)}, \tau^{(t-1)}\right) \end{array} \right]; \tag{5.9}$$

$$\pi^*_{\mathbf{R}} \in \arg\max_{\pi_{\mathbf{R}}} \mathbb{E}\left[\sum_t \gamma^t \mathcal{U}_\theta\left(s^{(t)}, a^{(t)}_{\mathbf{H}}, a^{(t)}_{\mathbf{R}}\right) \middle| \begin{array}{c} s^{(t+1)} \sim T\left(s^{(t)}, a^{(t)}_{\mathbf{H}}, a^{(t)}_{\mathbf{R}}\right); \\ a^{(t)}_{\mathbf{H}} \sim \pi^*_{\mathbf{H}}\left(s^{(t)}, \tau^{(t-1)}; \theta\right); \\ a^{(t)}_{\mathbf{R}} \sim \pi_{\mathbf{R}}\left(s^{(t)}, \tau^{(t-1)}\right) \end{array}\right]. \tag{5.10}$$

Note that pedagogic equilibrium need *not* maximize $\mathbb{E}[\mathcal{U}_\theta]$. Theorem 15 proves this by appealing to a counter-example apprenticeship game in the Lavaland domain from Section 4.2.

Theorem 15. *There exist* CIRL *games G with strategy pairs $(\pi_{\mathbf{H}}, \pi_{\mathbf{R}})$ such that*

1. *$(\pi_{\mathbf{H}}, \pi_{\mathbf{R}})$ is a pedagogic equilibrium in G; and*
2. *there exists an alternative strategy pair $(\pi^*_{\mathbf{H}}, \pi^*_{\mathbf{R}})$ that gets higher expected utility*

$$V^{\left(\pi^*_{\mathbf{H}}, \pi^*_{\mathbf{R}}\right)} > V^{(\pi_{\mathbf{H}}, \pi_{\mathbf{R}})}. \tag{5.11}$$

Proof. We show this by construction. Consider the Proof-of-Concept Lavaland domain from Section 4.2 where there are 4 types of states $\{s_{\text{grass}}, s_{\text{dirt}}, s_{\text{target}}, s_{\text{lava}}\}$ and $\phi(s)$ is a one-hot encoding of state type. Let G be an an apprenticeship game where the s_{lava} indicator is constant in the development environment $\tilde{E}$. Suppose, additionally, that there exist types θ_0, θ_1 such that

$$P_\theta(\theta_0) = P_\theta(\theta_1); \tag{5.12}$$

$$\mathcal{U}(s_{\text{lava}}, a_{\mathbf{H}}, a_{\mathbf{R}}; \theta_0) = -\mathcal{U}(s_{\text{lava}}, a_{\mathbf{H}}, a_{\mathbf{R}}; \theta_1); \tag{5.13}$$

$$\mathcal{U}(s, a_{\mathbf{H}}, a_{\mathbf{R}}; \theta_0) = \mathcal{U}(s, a_{\mathbf{H}}, a_{\mathbf{R}}; \theta_1), s \in \{s_{\text{grass}}, s_{\text{dirt}}, s_{\text{target}}\}; \tag{5.14}$$

$$\left|\arg\max_\tau \sum_{\left(s^{(t)}, a^{(t)}\right) \in \tau} \gamma^t \mathcal{U}\left(s^{(t)}, a^{(t)}_{\mathbf{H}}, a^{(t)}_{\mathbf{R}}; \theta_i\right)\right| > 1, i \in \{0, 1\}. \tag{5.15}$$

Let the deployment environment consist of two copies of $\tilde{E}$: one where the indicator for s_{lava} is on, and another where the indicator for s_{lava} is off. Call them $E_{\text{lava}}$and $E_{\neg\text{lava}}$. In addition, there is one new state s_0 with 2 actions a_{lava} and $a_{\neg\text{lava}}$. a_{lava} samples the next state from the initial distribution of E_{lava} and $a_{\neg\text{lava}}$ samples from the initial distribution of $E_{\neg\text{lava}}$.

We will now construct two pedagogic equilibria with distinct value, which will show the result. First consider the case where $\pi_{\mathbf{H}}$ maximizes immediate reward and breaks ties randomly. θ_0 and θ_1 will have the same likelihood in the posterior because they each induce the same distribution of utility evaluations in the development environment. Thus, Equation 5.12 and Equation 5.13 allow us to conclude that $Q_{\mathbf{R}}(s_0, a_{\text{lava}}; \mathcal{B}_{\mathbf{R}}) = Q_{\mathbf{R}}(s_0, a_{\neg\text{lava}}; \mathcal{B}_{\mathbf{R}})$ when $\mathbf{H}$ maximizes utility in $\tilde{E}$. As a result, $\mathbf{br}\left(\pi^\beta_{\mathbf{H}}\right)$ can take either action in s_0. Suppose $\mathbf{R}$ breaks ties randomly.

$\pi^{\beta}_{\mathbf{H}}$ *is* a best response to $\mathbf{br}\left(\pi^{\beta}_{\mathbf{H}}\right)$. There is nothing that $\mathbf{H}$ can do to change $\mathbf{R}$'s behavior in s_0, so $\mathbf{H}$ only has incentives to communicate the optimal trajectory — which $\pi^{\beta}_{\mathbf{H}}$ does with probability 1 as $\beta \to \infty$. Thus, for β sufficiently large, $\left(\pi^{\beta}_{\mathbf{H}}, \mathbf{br}\left(\pi^{\beta}_{\mathbf{H}}\right)\right)$ is a pedagogic equilibrium.

Now consider the case where $\mathbf{R}$ uses which optimal trajectory $\mathbf{H}$ chooses to break the tie between a_{lava} and $a_{\neg\text{lava}}$ as follows. Let $\tau_0 \neq \tau_1$ be distinct optimal trajectories for θ_0, θ_1. They exist by assumption due to Equation 5.15. Let $\mathbf{R}$ choose a_{lava} when $\mathbf{H}$ chooses τ_0 and $a_{\neg\text{lava}}$ when $\mathbf{H}$ chooses τ_1. Now, $\mathbf{H}$ *can* affect $\mathbf{R}$'s behavior in s_0, and so they have incentives to break ties in a way that will cause $\mathbf{R}$ to make the right choice. This pair of strategies is optimal: $\mathbf{H}$ executes the optimal trajectory for θ during development and $\mathbf{R}$ executes the optimal trajectory for both θ_0 *and* θ_1 during deployment.

Thus, this new pair of strategies is optimal for θ_0 and θ_1. This proves Equation 5.11 because we have constructed a CIRL game with (at least) two distinct pedagogic equilibria of different expected utility. □

Thus, we can have suboptimal strategy pairs that are mutual best-responses. Furthermore, the construction highlights the fact that even computing an optimal strategy pair is not the same as 'solving' a CIRL game for all intents and purposes. It is straightforward to construct multiple optimal strategy pairs where $\mathbf{R}$ interprets $\mathbf{H}$'s behavior in the *exact opposite way* with respect to θ_{lava}. Thus, the solution is not robust to *coordination failures*. Because we can not actually plan for $\mathbf{H}$—at best their behavior can be influenced indirectly—CIRL games must often be addressed under assumptions that reflect *decentralized planning* in addition to the decentralized execution constraints considered in the majority of the Dec-POMDP literature. Boutilier [22] explores these *coordination* challenges for the class of multi-agent MDPs. We discuss these issues (and general issues of brittle pedagogic equilibria in CIRL games) in depth in Chapter 6. For now, we will focus on the problem of efficiently computing optimal pedagogic equilibria.

In the optimal pedagogic equilibrium, $\mathbf{R} \circ \mathbf{H}$ is optimized to maximize *team* performance. As a result, $\mathbf{H}$ uses all available information to optimally balance the long-term value of information that their actions reveal to $\mathbf{R}$ and the short-term costs of deviating from the myopically optimal action. The problem of computing an optimal pedagogic equilibrium for a CIRL game can be reduced to solving a *decentralized*-POMDP(Dec-POMDP) [18]. A Dec-POMDP extends a POMDP to have multiple actors, each with potentially distinct observation distributions. Most Dec-POMDP research deals with (offline, centralized) planning algorithms that determine policies for all the actors. Each policy is restricted to depend only on the observation-action history of the appropriate actor. In other words, the goal is to identify a set of policies that maximize utility and can be executed independently, each relying only on its own observations. Actions can be coordinated only, but only through changes to the world state that the other actors can observe. In a general Dec-POMDP, the actors must generally track a belief about the others' information set [18]. The requirement to maintain *meta-beliefs* (i.e., a belief about the other actors' beliefs) accounts for the (dou-

bly exponential) NEXP-complete time complexity of Dec-POMDP solution algorithms [18]. In the next section, we show that the commonly observed nature of s avoids this complexity.

$\mathcal{B}_{\mathbf{R}}$ is a sufficient statistic for optimal policy pairs

Nayyar, Mahajan, and Teneketzis [114] shows that a Dec-POMDP can be reduced to a *coordination*-POMDP. The actor in this POMDP is a coordinator that observes all common observations and specifies a local policy for each actor called a decision rule. These policies map each actor's private information to an action. The structure of a CIRL game implies that the private information is limited to **H**'s initial observation of θ. This allows the reduction to a coordination-POMDP to preserve the size of the (hidden) state space, making the problem easier.

Theorem 16. *Let M be an arbitrary* CIRL *game with state space $\mathcal{S}$ and reward space Θ. There exists a (single-actor)* POMDP *$M_{\mathbf{C}}$ with (hidden) state space $\mathcal{S}_{\mathbf{C}}$ such that $|\mathcal{S}_{\mathbf{C}}| = |\mathcal{S}| \cdot |\Theta|$ and, for any policy pair in M, there is a policy in $M_{\mathbf{C}}$ that achieves the same sum of discounted rewards.*

Proof. We take $M_{\mathbf{C}}$ to be the coordination POMDP associated associated with M. The second component of **C**'s action is an action for **R**. **R** has no private observations, so for any policy $\pi_{\mathbf{R}}$ **R** could choose to follow, **C** can match it by simulating $\pi_{\mathbf{R}}$ and outputting the corresponding action. Similarly, **C** only observes common observations, so **R** can implement any coordinator strategy by simulating **C** and directly executing the appropriate action.

By a similar argument, **H** can also simulate any given $\pi_{\mathbf{C}}$ to compute their decision rule $v_{\mathbf{H}}$, and then execute the corresponding action. To see that there is a $\pi_{\mathbf{C}}$ that can reproduce the behavior of any $\pi_{\mathbf{H}}$, let h be the action-observation history for **H**. **C** can choose the following decision rule

$$v_{\mathbf{H}}(\theta) = \pi_{\mathbf{H}}(\theta; h)$$

to produce the same behavior. □

An immediate consequence of this result is that **R**'s belief about θ is a sufficient statistic for optimal behavior.

Theorem 17. *Let M be a* CIRL *game. There exist optimal policies $(\pi^*_{\mathbf{H}}, \pi^*_{\mathbf{R}})$ that only depend on the current state and* **R***'s belief.*

$$\pi_{\mathbf{H}}{}^* : \mathcal{S} \times \Delta(\Theta) \times \Theta \to \mathcal{A}_{\mathbf{H}}, \qquad \pi_{\mathbf{R}}{}^* : \mathcal{S} \times \Delta(\Theta) \to \mathcal{A}_{\mathbf{R}}.$$

Proof. [149] showed that an optimal policy in a POMDP only depends on the belief state. **R**'s belief uniquely determines the belief for **C**. From this, an appeal to Theorem 16 shows the result. □

Remark 10. *In a general Dec-*PoMdp*, the hidden state for the coordinator-*PoMdp *includes each actor's history of observations. In* Cirl*, θ is the only private information so we get an exponential decrease in the complexity of the reduced problem. This allows one to apply general* PoMdp *algorithms to compute optimal joint policies in* Cirl*.*

It is important to note that the reduced problem may still be very challenging. PoMdps are difficult in their own right and the reduced problem still has a much larger action space. That being said, this reduction is still useful in that it characterizes optimal joint policy computation for Cirl as significantly easier than Dec-PoMdps. Furthermore, this theorem can be used to justify approximate methods (e.g., iterated best response) that only depend on **R**'s belief state.

A efficient dynamic programming algorithm for CIRL

Next, we will build on this result to design efficient (approximate) dynamic programming method for Cirl games. The previous reduction greatly simplifies the belief space that must be considered during (joint) policy optimization. However, this comes at the cost of much larger action space. This is because the action space in the coordination PoMdp is the product of **R**'s action space $\mathcal{A}_{\mathbf{R}}$ and the set of mapping from **H**'s type to their action space. This leads to an action space in the coordinator PoMdp with $|\mathcal{A}_{\mathbf{R}}| \cdot |\mathcal{A}_{\mathbf{H}}|^{|\Theta|}$ distinct actions to be considered. To counteract this blowup, we can take advantage of the special information structure of a Cirl game. **H** is a full-information actor and so, although their policy is not Markov with respect to the environment state, it can be computed exactly from intermediate calculations already used in PoMdp value iteration.

A cooking assistance game

We will introduce our method with the use of an example cooking assistance game. In this game, **R** and **H** are working together to make food. There are several ingredients that can be prepared to cook recipes. **H**'s type space Θ is the set of possible recipes.

Take for instance the domain from Figure 5.6. **H** and **R** work to prepare a meal using three ingredient types: bread, meat, and tomatoes. **H** wants to prepare either a sandwich (2 bread, 1 meat, 0 tomatoes), or tomato soup (1 bread, 1 meat, 2 tomatoes). **R** does not know *a priori* which meal **H** wants. At every time step, **R** and **H** each prepare a single unit of any ingredient, or no ingredient at all. The state produces utility 1 if the desired recipe has been made and 0 otherwise. In this domain, (optimal) pedagogic equilibrium models **H**'s incentive for the robot to infer the correct recipe. The structure of Θ allows **R** to act, at least initially, based on incomplete information about the correct recipe: bread and meat are a good idea for either recipe.

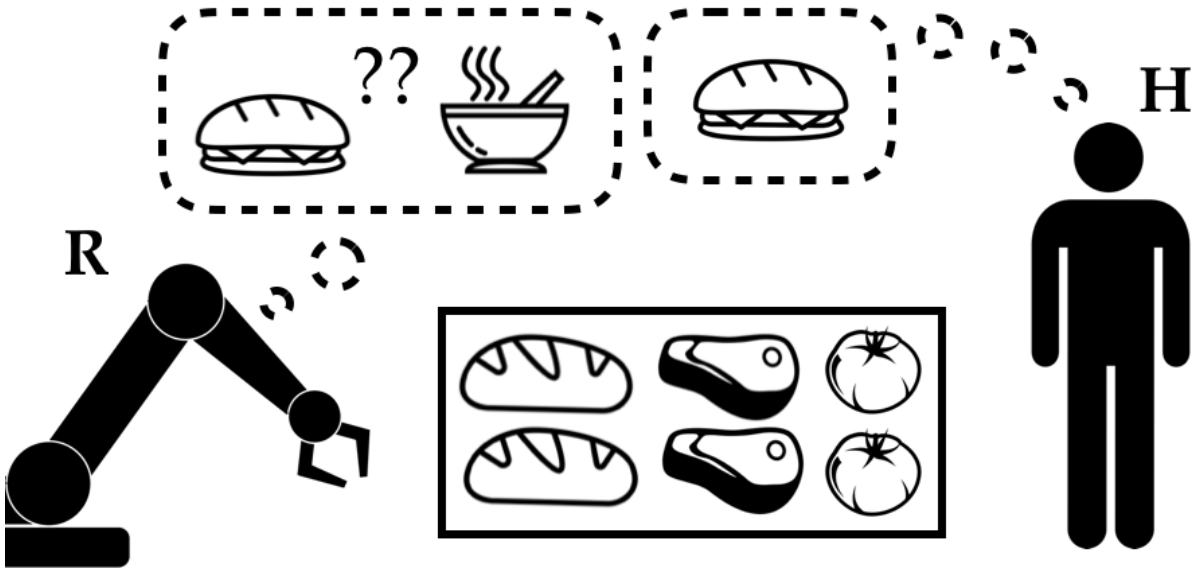


Figure 5.6: An example of the cooking domain. **H** and **R** need to work together to prepare a meal using **H**'s desired recipe. **R** starts off unaware of which meal **H** wants. In our running example, there are 3 types of ingredients: bread, meat, and tomatoes. There are 2 recipes: a sandwich (2 bread, 1 meat, 0 tomatoes) and tomato soup (1 bread, 1 meat, 2 tomatoes). The world state represents how many (0, 1, or 2) of each ingredient has been prepared. In a *pedagogic* solution for this game, **H** chooses to make tomatoes first if they want tomato soup; the other ingredients are needed for both recipes, so **R** can make either. **R** can then infer that **H** wants a sandwich if they make bread, because they would have made tomatoes otherwise. This pragmatic reasoning on **R**'s part is a distinguishing property of pedagogic equilibrium in assistance games. No IRL-**R** can make this inference because bread is needed for both recipes.

Cooking as an Assistance-POMDP

Before introducing our approach, we describe value iteration as it applies to assistance-POMDPs. As discussed in Chapter 2, value iteration algorithms often track a set of potentially optimal *conditional plans*. We write a conditional plan as $\sigma = \left(a_{\mathbf{R}}^{(t)}, v^{(t)}\right)$. $a_{\mathbf{R}}^{(t)} \in \mathcal{A}_{\mathbf{R}}$ indicates **R**'s next action. $v^{(t)}$ maps the current observation, **H**'s action $a_{\mathbf{H}}^{(t)}$, into the next conditional plan.

The value of a conditional plan depends on the associated belief state $\mathcal{B}_{\mathbf{R}}$. A useful observation for POMDP algorithms is that this can be represented efficiently as a vector of values: one for each possible hidden state. In an assistance-POMDP, this is a mapping from

Θ the value of the given conditional plan when $\mathbf{H}$ has that type.

The α-vector of a conditional plan $\sigma = (a_{\mathbf{R}}, v)$ in a assistance-PoMdp, is defined by the following dynamic programming relationship:

$$\alpha^{\sigma}(s;\theta) = \mathbb{E}\left[\mathcal{U}(s, a_{\mathbf{H}}, a_{\mathbf{R}};\theta) + \gamma \sum_{s' \in \mathcal{S}} T(s' \,|s, a_{\mathbf{H}}, a_{\mathbf{R}})\, \alpha^{v(a_{\mathbf{H}})}(s';\theta) \,\middle|\, a_{\mathbf{H}} \sim \pi_{\mathbf{H}}(s;\theta)\right]. \quad (5.16)$$

The value of a plan at a belief $\mathcal{B}_{\mathbf{R}}$ is the expected value of the plan across the states i.e. $V^{\sigma}(\mathcal{B}_{\mathbf{R}}) = \mathcal{B}_{\mathbf{R}} \cdot \alpha^{\sigma} = \sum_{s \in \mathcal{S}} \mathcal{B}_{\mathbf{R}}(s) \cdot \alpha^{\sigma}(s)$. The goal of an agent in a PoMdp is to find the plan with maximal value from their current belief.

Value iteration [153] can be used to compute the optimal conditional plan. This algorithm starts at the horizon and works backwards. It generates new conditional plans at each time-step and evaluates them according to Equation 5.16. It constructs all potentially optimal plans of a given length, set in relation to γ to ensure optimality, and selects the one with maximal value at the initial belief.

Solving an assistance-PoMdp for cooking. For example, consider a simplified instance of the cooking task from Figure 5.6 where $\mathbf{H}$ picks their actions according to only their desired recipe and the quantity of each ingredient prepared so far, i.e., they do not consider $\mathbf{R}$'s past or future behavior when picking their actions. The simplified cooking task is now an assistance-PoMdp. I.e., $\mathbf{H}$ is modeled as a Markov component of the environment.

$\mathcal{B}_{\mathbf{R}}$ is a distribution over possible recipes, in this case over the set {Sandwich, TomatoSoup}. $\mathbf{R}$'s behavior is specified by a conditional plan $\sigma = (a, v)$. For example,

$$\sigma = \left(\text{Bread}, \quad \begin{cases} \text{Bread} \to (\text{Meat}, \emptyset) & \#\# \text{ Target recipe: Sandwich} \\ \text{Tomatoes} \to (\text{Meat}, \text{Tomatoes}) & \#\# \text{ Target recipe: TomatoSoup} \\ \text{Meat} \to (\emptyset, v') & \#\# \text{ Target recipe: ??} \end{cases}\right). \quad (5.17)$$

This is one component of a conditional plan for an Irl-$\mathbf{R}$. $\mathbf{R}$ prepares bread first, because it is needed for both recipes. If $\mathbf{H}$ makes bread, then we know the desired recipe is a sandwich. $\mathbf{R}$ prepares the final piece of bread to complete the sandwich. If $\mathbf{H}$ makes tomatoes, then the recipe is tomato soup, and $\mathbf{R} \circ \mathbf{H}$ finishes the recipe in the next round. However, if $\mathbf{H}$ responds by preparing meat, θ is not identified. $\mathbf{R}$ must make a guess at the next action, potentially wasting food, time, or both. In this case, recipe completion proceeds according to the decision rule v'.

Note that σ *is* a best response to $\pi_{\mathbf{H}}$ in this example and $\pi_{\mathbf{H}}$ chooses actions that are optimal for $\mathbf{H}$ in isolation. Thus, Equation 5.17 is an optimal solution to the assistance-PoMdp formulation of the cooking problem. The ultimate value of the policy comes down to details of how $\mathbf{H}$ breaks ties between optimal-in-isolation actions. If $\mathbf{H}$ would simply adopt the policy

$$v^{\text{Pedagogy}} = \{\text{Sandwich} \to \text{Bread}; \text{TomatoSoup} \to \text{Tomatoes}\}, \quad (5.18)$$

then the target recipe is guaranteed to be made in two rounds (as opposed to 3). In the next section, we will show how Cirl model's $\mathbf{H}$'s incentives to make this change.

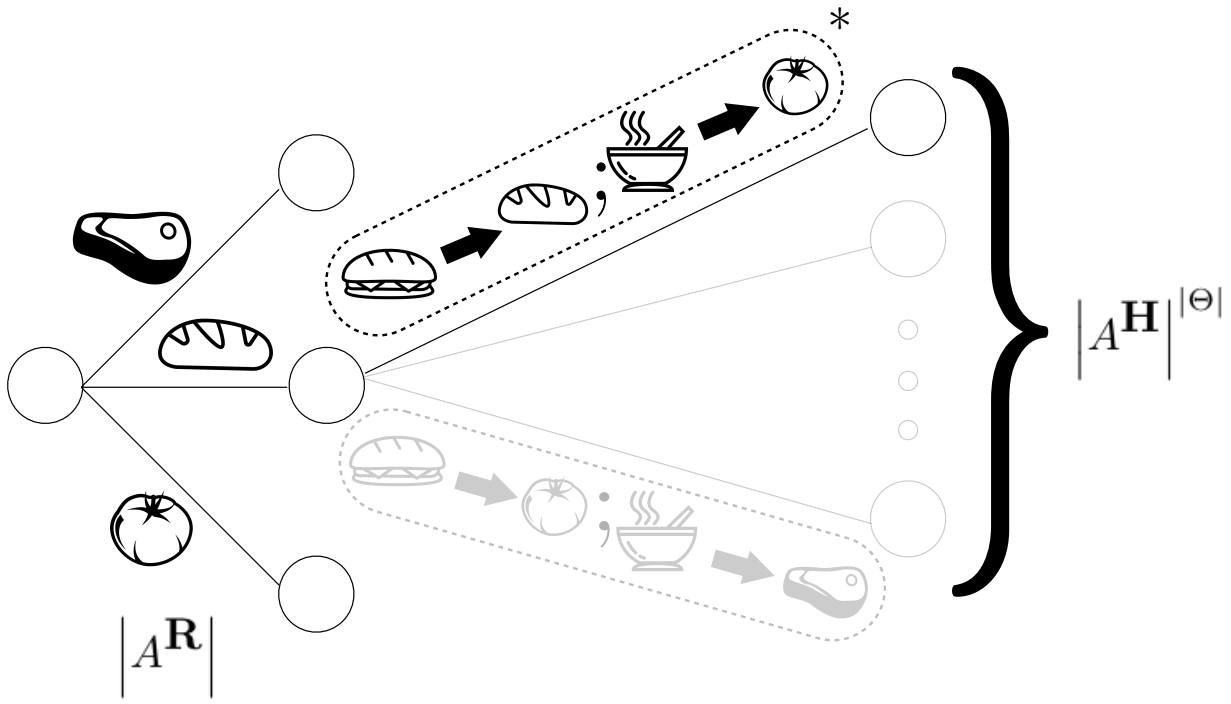


Figure 5.7: A node in the search tree from the POMDP reduction of the ChefWorld domain. Actions are tuples that contain an action for $\mathbf{R}$ and a decision rule for $\mathbf{H}$ – a mapping from their desired recipe to an action. This leads to a branching factor of $|\mathcal{A}_{\mathbf{H}}|^{|\Theta|}|\mathcal{A}_{\mathbf{R}}|$ and makes application of POMDP methods inefficient. The modified Bellman update from Section 5.2 prunes away all of $\mathbf{H}$'s decision rules but the optimal response. (In the diagram, the gray branches are pruned away by the modified Bellman update.)

The Coordination-POMDP for cooking

Now, consider what changes when we model the problem as a CIRL game. $\mathbf{H}$'s policy can now account for $\mathbf{R}$'s conditional plan σ. $\mathbf{H}$ can select actions in response to Q_θ^σ. Thus, they recognize the suboptimality of preparing meat initially. This causes them to follow Equation 5.18, increasing the expected utility from the interaction.

However, this comes at a computational cost. Consider the coordination-POMDP reduction for this example. The action space has become much larger. It must specify an action for $\mathbf{R}$ $a_{\mathbf{R}}$ and a decision rule $v_{\mathbf{H}}$ for $\mathbf{H}$. The size of the action space for the cooking assistance-POMDP is 4: one action for each ingredient and an additional no-op, $\emptyset$. The action space in the coordination-POMDP increases this to $32 = 2^5$: there are $|\mathcal{A}_{\mathbf{H}}|^{|\Theta|} = 4^2$ distinct decision rules for $\mathbf{H}$ which can each be paired with a different $a_{\mathbf{R}}$. Figure 5.7 shows an illustration of this action space as a tree.

The computational efficiency of an assistance-POMDP formulation (and the associated approximation) is clear. **H**'s component of the action can be marginalized over if they are modeled as a Markov observation model. The key insight we leverage in our approach is that it is still possible to do this marginalization *even if* **H***'s behavior depends on* V_θ^σ.

Theorem 18. *Let* $G = \langle \{\mathcal{S}, P_\mathcal{S}\}, \{\mathcal{A}_\mathbf{R}, \mathcal{A}_\mathbf{H}\}, T, \{\Theta, P_\theta\}, U \rangle$ *be a* CIRL *game.* **H***'s value function for a conditional plan* σ *is equal to* α^σ*. That is, for any conditional plan* σ *and* $\theta \in \Theta$

$$V_\theta^\sigma(s) = \alpha^\sigma(s; \theta). \tag{5.19}$$

Proof. This follows directly from the definition of α^σ as the vector of values for σ as a function of the latent state (s, θ). □

In a general Dec-POMDP computing the value function for any actor requires taking a dot product with their belief. However, full-information actors have a very restricted class of possible beliefs: beliefs that place all their mass on a single state. Thus, α^σ tracks the value function of full-information actors. This means that fully-informed, cooperative, strategic behavior can be accounted for with minimal increase in computational complexity.

Specifically, we can update the dynamic programming relationship in Equation 5.16 to account for **H**'s strategic behavior by maximizing over $a_\mathbf{H}$:

$$\alpha^\sigma(s; \theta) = \max_{a_\mathbf{H} \in \mathcal{A}_\mathbf{H}} \left\{ \mathcal{U}(s, a_\mathbf{H}, a_\mathbf{R}; \theta) + \gamma \mathbb{E}\left[\alpha^{v(a_\mathbf{H})}(s'; \theta) \middle| s' \sim T(s, a_\mathbf{H}, a_\mathbf{R}) \right] \right\}. \tag{5.20}$$

Corollary 18.1. *For any* CIRL *game* G*, the policy computed by value iteration with Equation 5.20,* $\pi_\mathbf{R}^*$ *is a component of an optimal pedagogic equilibrium* $(\pi_\mathbf{H}^*, \pi_\mathbf{R}^*)$.

Proof. From Theorem 18, we know that **H**'s value function for a given conditional plan σ is given by α^σ. Thus, the maximization in Equation 5.20 never prunes away an optimal action for **H**. Thus, the conditional plan output at the end of value iteration, which is the best policy among those not pruned away, must be optimal. □

This modification to the Bellman update allows us to solve a CIRL game without having to include the set of **H**'s decision rules in the action space. As depicted in Figure 5.7, the modified Bellman update computes **H**'s optimal action given the current state and the robot's plan; all of **H**'s other actions are pruned away in the search tree. The size of the action space is then $|\mathcal{A}_\mathbf{R}|$ instead of $|\mathcal{A}_\mathbf{H}|^{|\Theta|}|\mathcal{A}_\mathbf{R}|$. POMDP algorithms are exponential in the size of the action space; this modification therefore allows us to solve CIRL games much more efficiently. The following theorem establishes the associated complexity gains.

Theorem 19. *The modification to the Bellman update presented above reduces the time and space complexity of a single step of value iteration by a factor of* $\mathcal{O}\left(|\mathcal{A}_\mathbf{H}|^{|\Theta|}\right)$.

Algorithm 1 Adapted Value Iteration for CIRL Games

procedure ValueIteration
  $\Gamma_t \leftarrow$ Set of trivial plans
  for $t \in \{T-1, T-2, \ldots, 1, 0\}$ **do**
    $\Gamma_{t+1} \leftarrow \Gamma_t$
    $\Gamma_t \leftarrow$ Set of all plans beginning at time t
    for $\sigma \in \Gamma_t$ **do**
      for $s = (x, \theta) \in S$ **do**
        $Q_H(s, a^H, \sigma) = \sum_{s'} T(s, a^H, a^R, s') \cdot \alpha_{v(a^H)}(s')$
        $\alpha_\sigma(s) = R(s) + \gamma \cdot \sum_{a^H} \pi_H(a^H \mid Q_H(s, a^H, \sigma)) \cdot Q_H(s, a^H, \sigma)$
    $\Gamma_t \leftarrow \text{Prune}(\Gamma_t)$
  $a^R_* = \arg\max_{\sigma \in \Gamma_0} \alpha_\sigma \cdot b_0$
  Return a^R_*

Proof. The complexity of one step of POMDP value iteration is linear in the size of the action space, $|A|$ [129]. Since the structure of our algorithm is identical to that of exact value iteration, this is also true for our algorithm.

The action space in the POMDP reduction of CIRL has size $|\mathcal{A}_{\mathbf{H}}|^{|\Theta|}|\mathcal{A}_{\mathbf{R}}|$. Our modification to the Bellman update reduces the size of the action space to simply $|\mathcal{A}_{\mathbf{R}}|$. Therefore, our algorithm reduces the time taken to run, and number of plans generated at, each time step by a factor of $|\mathcal{A}_{\mathbf{H}}|^{|\Theta|}$. □

Our experimental domain is based on the cooking example. Assume there are m recipes and n ingredients. The state space is an n-tuple representing the quantity of each ingredient prepared thus far. At each time step, each agent can prepare any of the n ingredients or none at all. Each of the m recipes corresponds to a different θ (i.e. reward parameter) value. Both agents receive a reward of 1 if **H**'s desired recipe is prepared correctly and a reward of 0 otherwise. The robot **R** begins the game entirely uncertain about **H**'s desired recipe i.e. **R** has a uniform belief over Θ. In our first experiment, we compared the time taken by exact VI and by our adaptation of it with the modified Bellman update. We first fixed the number of ingredients at two and varied the number of recipes in the domain. Table 5.2 compares the results. For the simpler problems, where the number of recipes was 2 or 3, our adapted algorithm solved the problem up to ∼3500× faster than exact VI. On more complex problems where the number of recipes is greater than 3, exact VI failed to solve the problem after depleting our system's 16GB memory; in contrast, our adapted algorithm solved each of these more complex problems in less than 0.5 seconds. We next fixed the number of recipes and compared the performance of both these algorithms for various numbers of ingredients. Both the exact methods, but especially the one using the standard Bellman update, scaled much worse with the number of ingredients than with the number of recipes. With even three ingredients, exact VI timed out and failed to solve the problem within two hours; our algorithm however solved the problem in five seconds.

Table 5.2: Time taken (s) to find the optimal robot policy using exact VI and our adaptation of it for various numbers of possible recipes. NA denotes that the algorithm failed to solve the problem.

# Recipes	Exact VI	Ours
2	4.448 ± 0.057	0.071 ± 0.013
3	394.546 ± 6.396	0.111 ± 0.013
4	NA	0.158 ± 0.003
5	NA	0.219 ± 0.007
6	NA	0.307 ± 0.005

Comparing CIRL and IRL solutions

Next, we consider what advantages CIRL has compared to IRL. On a collaborative task, IRL is equivalent to assuming that **H** chooses their actions in isolation, and **R** uses observations of **H**'s behavior to infer their preferences. Specifically, **H** solves a single-agent, fully-observable, variant of the problem, and **R** responds by solving the appropriate assistance-POMDP.

We fix the number of ingredients at 3 and vary the number of recipes. Figure 5.8 shows the results. In each experiment the optimal CIRL solution prepares the correct recipe while the IRL solution fails to do so up to 50% of the time. To understand the nature of this difference in performance, we analyze the CIRL and IRL solutions. Consider a case of our running example from before with two recipes. The state is a tuple $(\#meat, \#bread, \#tomatoes)$ and $\Theta = \{sandwich = (1, 2, 0), soup = (1, 1, 2)\}$. For both approaches, **R** initially prepares meat. In the baseline IRL solution, **H** can initially make any ingredient if they want soup and can make meat or bread if they wants a sandwich. In each case, **H** chooses uniformly at random between allowed ingredients. This conveys some information about their desired recipe, but is not enough to uniquely identify it. Since the same ingredient is optimal for multiple recipes, **R** is still confused after one turn. This means **R** will sometimes fail to complete the desired recipe, reducing average utility.

The CIRL solution, in contrast, relies on the implicit communication between the human and the robot. Here, if **H** wants soup, she prepares tomatoes, as opposed to any ingredients that are common with the sandwich. Even more interestingly, she waits (i.e. does nothing) if she wants a sandwich. This is pure signaling behavior—*waiting is suboptimal in isolation*, but picking an ingredient is more likely to confuse the robot. In turn **R** knows that **H** would have picked tomatoes if she wanted soup, and responds appropriately.

In other words, **H** teaches the robot about her preferences with her action selection. This works because **H** knows that **R** will interpret her behavior pragmatically, i.e., **R** expects to be taught by **H**. This is reflected in the experiment: the optimal CIRL solution prepares the correct recipe each time.

The value alignment problem is necessarily cooperative: without the robot, the human is

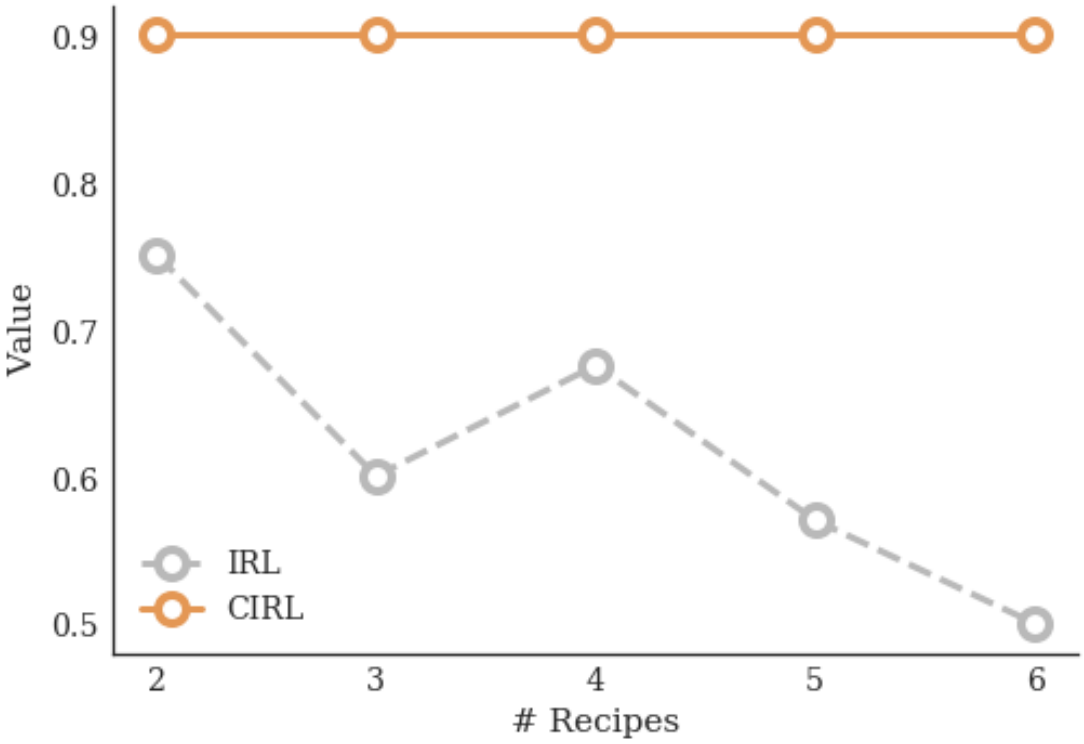


Figure 5.8: Value attained by CIRL and standard IRL on the cooking domain with various numbers of possible recipes. Unlike IRL, CIRL produces solutions where **H** picks their actions pedagogically and **R** reasons about **H** accordingly.

unable to complete her desired task, and without explicit signaling from the human, the robot learns inefficiently, is less valuable and is more likely to make a mistake. Pedagogic behavior from **H** naturally falls out of the CIRL solution. In response, **R** interprets **H**'s actions pragmatically. These instructive and communicative behaviors allow for faster learning and create an opportunity to generate higher value for the human.

Efficient Approximate Algorithms for CIRL

Adapting Point-Based Value Iteration

Background Point Based Value Iteration (PBVI) is an approximate algorithm used to solve POMDPs [121]. The algorithm maintains a representative set of points in belief space and an α-vector at each of these belief points. It performs approximate value backups at each of

these belief points using this set of α-vectors. Let Γ_{t+1} denote the set of α-vectors for plans that begin at time $t+1$. The value at time t at a belief b is approximated as:

$$V(b) = \max_{a \in A} \left[\sum_{s \in S} R(s)b(s) + \gamma \sum_{o \in O} \max_{\alpha \in \Gamma_{t+1}} \sum_{s \in S} \left(\sum_{s' \in S} P(s', o \mid s, a)\alpha(s) \right) b(s) \right]. \tag{5.21}$$

The algorithm trades off computation time and solution quality by expanding the set of belief points over time: it randomly simulates forward trajectories in the POMDP to produce new, reachable beliefs.

Our Adaptation If $\mathbf{R}$ takes action $a_{\mathbf{R}}$ and follows a conditional plan σ, then $\mathbf{H}$'s Q-values are $Q_{\mathbf{H}}(x, a_{\mathbf{H}}, a_{\mathbf{R}}, \alpha) = \sum_{s'} T(s, a_{\mathbf{H}}, a_{\mathbf{R}}, s') \cdot \alpha_\sigma(s')$. Notice that we can compute these Q-values at each step of PBVI. This lets us use the modified Bellman update and to adapt PBVI to solve CIRL games specifically. We replace the transition-observation distribution in the PBVI backup rule with

$$P(s', a_{\mathbf{H}} \mid s, a_{\mathbf{R}}, \alpha) = T(s, a_{\mathbf{H}}, a_{\mathbf{R}}, s') \cdot \pi_{\mathbf{H}}(Q_{\mathbf{H}}(x, a_{\mathbf{H}}, a_{\mathbf{R}}, \alpha)). \tag{5.22}$$

The modified backup rule for PBVI is thus given by

$$V(b) = \max_{a_{\mathbf{R}} \in \mathcal{A}_{\mathbf{R}}} \left[\sum_{s \in S} R(s)b(s) + \gamma \sum_{a_{\mathbf{H}}} \max_{\alpha \in \Gamma_{t+1}} \sum_{s \in S} \left(\sum_{s' \in S} P(s', a_{\mathbf{H}} \mid s, a_{\mathbf{R}}, \alpha)\alpha(s) \right) b(s) \right]. \tag{5.23}$$

We now show that the approximate value function in PBVI converges to the true value function. Let ϵ_B denote the density of the set of belief points B in PBVI. Formally, $\epsilon_B = \max_{b \in \Delta} \min_{b' \in B} ||b - b'||_1$ is the maximum distance from any reachable, legal belief to the set B.

Theorem 20. *For any belief set B and horizon n, the error of our adapted PBVI algorithm $\eta = ||V_n - V_n^*||_\infty$ is bounded as*

$$\eta \leq \frac{(R_{\max} - R_{\min})\epsilon_B}{(1-\gamma)^2}.$$

Proof. Since the dynamics of our problem are now time-varying instead of static, the backup operator applied at every time-step changes. Let H_t denote the backup operator applied to compute the value of V_t. It will suffice to show that each such backup operator H_t is a contraction mapping. The result then follows by following the proof of Theorem 1 in [121] exactly. We will prove the result by showing that each H_t is the backup operator for a specific POMDP and thus, for this POMDP's corresponding belief MDP; it must therefore be a contraction mapping.

Take H_t for some $1 \leq t \leq n$. We will now construct a new POMDP for which H_t is the backup operator. Let $\hat{S} = S \times \Gamma_{t+1}$, where Γ_{t+1} denotes the set of α-vectors from our original problem at time $t+1$. Let the α-vector component of the state be static i.e.

$P((s', \alpha') \mid (s, \alpha), a_{\mathbf{R}}) = 0$ if $\alpha \neq \alpha'$. The action and observation spaces remain as they are in our original problem. The transition-observation distribution, given in Eqn. (3), is now time-invariant: we do not need to look forward in the search tree to compute the Q-values since the α-vectors are available in the state space. Hence, this POMDP is well-defined.

The dynamics of the POMDP are static and identical to the dynamics of our problem at time t. Thus, the backup operator H_t is the backup operator for this POMDP and also for this POMDP's corresponding belief MDP. Therefore, the backup operator H_t must be a contraction mapping. □

Next, we compared the values attained by PBVI and our adaptation, with one hour of computation time. We first fixed the number of recipes and varied the number of ingredients. The results of this experiment are presented in Figure 5.9. We found that both these algorithms, but especially our adapted algorithm, scaled much better with the number of ingredients than their exact VI counterparts. For simpler games, with 3 and 4 ingredients, both algorithms attained the maximal value of 0.9025. However, with 5 ingredients, PBVI found a value of 0 in an hour. In contrast, our algorithm easily solved the game with 5, 6, and 7 ingredients, attaining the maximal value of 0.9025. We next fixed the number of ingredients and varied the number of recipes. Again, our adapted algorithm outperformed PBVI. For example, with 4 recipes, our adapted method attains a value of 0.67875 while the standard PBVI method attains a value of 0.45125.

These results suggest that our modified Bellman update allows PBVI to scale to larger CIRL games, especially in terms of the size of **H**'s and **R**'s action space. This offers further support to our hypothesis.

Adapting Partially-Observed Monte-Carlo Planning

Background POMCP is a Monte Carlo tree-search (MCTS) based approximate algorithm for solving large POMDPs [143]. The algorithm constructs a search tree of action-observation histories and uses Monte Carlo simulations to estimate the value of each node in the tree. During search, actions within the tree are selected by UCB1. This maintains a balance between exploiting actions known to have good return and exploring actions not yet taken [89]. At leaf nodes, a rollout policy accrues reward which is then backed up through the tree. The algorithm estimates the belief at each node by keeping track of the hidden state from previous rollouts.

POMCP scales well with the size of the state space, but not with the size of the action space, which determines the branching factor in the search tree. POMCP is thus ill-suited to solving the reduced POMDP of CIRL games since the size of the action space is $|\mathcal{A}_{\mathbf{H}}|^{|\Theta|}|\mathcal{A}_{\mathbf{R}}|$.
Our Adaptation Using the idea behind our modified Bellman update, we adapt POMCP to solve CIRL games more efficiently. We approximate **H**'s policy while running the algorithm (much like we exactly compute **H**'s policy in exact value iteration). We maintain a live estimate of the sampled Q-values for **H** at each node. With enough exploration of the search tree (for instance, if actions are selected using UCB1), the estimated Q-values converge to

Algorithm 2 Adapted PBVI for CIRL Games

procedure PBVI(b_0, T)
  $B \leftarrow \{b_0\}$
  $V \leftarrow$ Set of α-vectors belonging to trivial plans
  repeat
    for $t \in \{T-1, T-2, \ldots, 1, 0\}$ **do**
      $V \leftarrow \text{Backup}(B, V)$
    $B \leftarrow \text{Expand}(B, V)$
  until $\max_{\alpha \in V} \alpha \cdot b_0 \geq V_{target}$
  return V

procedure BACKUP(B, V')
  $V \leftarrow \{\}$
  for $a^R \in \mathcal{A}_{\mathbf{R}}$ **do**
    for $\alpha_i' \in V'$ **do**
      for $s \in S$ **do**
        $Q_H(s, a^H) = \sum_{s'} T(s, a^H, a^R, s') \cdot$
          $\alpha_{v(a^H)}(s')$
      $\Gamma^{a^R} \leftarrow \alpha_i(s) = r(s) + \gamma \cdot$
        $\sum_{a^H} \pi_H(a^H \mid Q_H(s, a^H)) \cdot$
          $\sum_{s'} P(s', a^H \mid s, a^R, \alpha_i') \cdot \alpha_i(s')$
  for $b \in B$ **do**
    $V_b \leftarrow \{\}$
    for $a^R \in \mathcal{A}_{\mathbf{R}}$ **do**
      $V_b \leftarrow argmax_{\alpha \in \Gamma^{a^R}} \alpha \cdot b$
    $V \leftarrow argmax_{\alpha \in \Gamma_b} \alpha \cdot b$
  return V

procedure EXPAND(B', V')
  $B \leftarrow B'$
  for $b, \alpha \in B', V'$ **do**
    $B_b \leftarrow \{\}$
    for $a^R \in A^R$ **do**
      $s \sim b(s)$
      $a^H \sim P(a^H \mid s, a^R, \alpha)$
      $b'(s') = \eta \sum_s P(s', a^H \mid s, a^R, \alpha) b(s)$
        where η is the normalizing constant
      $B_b \leftarrow B_b \cup b'$
    $B \leftarrow B \cup argmax ||b - b'||_1, \forall b \in B_b, b' \in B'$
  return B

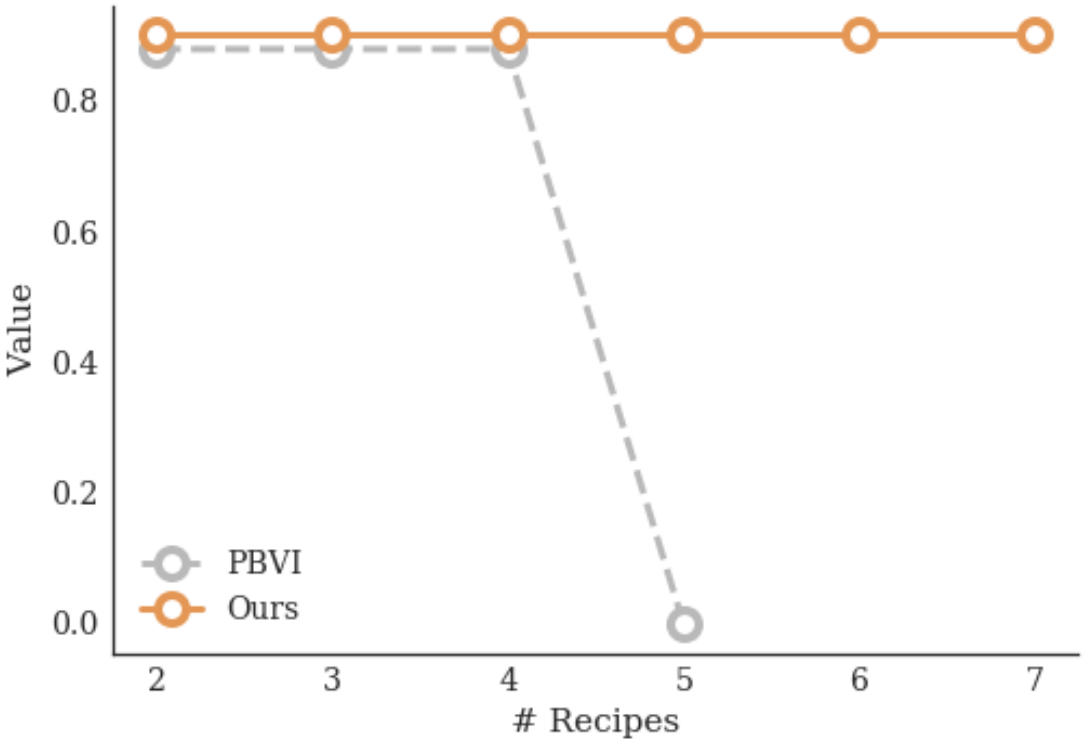


Figure 5.9: Value attained by PBVI and our approximate algorithm for various numbers of ingredients. (For the first 3 data points, the values attained by both methods were the same – we jittered the plot slightly for visibility).

the true values (in the limit). This guarantees that **H**'s policy converges to her true policy. The following result establishes convergence of our algorithm.

Theorem 21. *With suitable exploration, the value function constructed by our adapted POMCP algorithm converges in probability to the optimal value function, $V(h) \to V^*(h)$. As the number of visits $N(h)$ approaches infinity, the bias of the value function $\mathbb{E}[V(h) - V^*(h)]$ is $O(log(N(h))/N(h))$.*

Proof. We will show that with enough exploration, in the limit of infinite samples, we have a well defined POMDP. The result then follows from Theorem 1 in [143].

The human action nodes in the search tree maintain an array of values, which store the values of picking that action for different θ. At any point in the search tree, the human actions

(like the robot actions) are selected by picking the one that has the maximum augmented value (current estimated value plus exploration bonus). So, in the limit of infinite samples, each human action node is visited infinitely many times and the value estimates at the nodes converge to the true Q values. Having the correct human Q values gives us a POMDP, with well defined transition-observation dynamics. The result then follows from applying the analysis given in Theorem 1 of [143] to this POMDP. □

We compared the value attained in 30,000 samples by POMCP and by our adaptation with the modified Bellman update. We additionally compared these algorithms with FV-POMCP, a state-of-the-art MCTS method for solving MPOMDPs, a type of Dec-POMDP in which all agents can observe each others' behavior (as in CIRL).

We first fixed the number of recipes at 2 and varied the number of ingredients. Our adapted algorithm outperformed the other two algorithms across the board, especially for large numbers of ingredients. The results of this comparison are presented in Figure 5.10. POMCP did poorly on games with more than 4 ingredients. Although FV-POMCP scaled better to more complex games than POMCP, its values had high variance. For the largest games, with 6 and 7 ingredients, our adapted algorithm was the only one capable of solving the problem in 30,000 iterations. We also compared the value attained by each algorithm across 500,000 samples on the 6 ingredient game. The results of this comparison are depicted in Figure 5.11. Our algorithm converged to the true value faster than either of the other algorithms.

We next fixed the number of ingredients at 4 and varied the number of recipes. We found that the results of this experiment broadly matched the results of our previous experiment where we varied the number of ingredients. For example, with 4 recipes, our method achieves an average value of 0.631 ± 0.221 in 30,000 iterations while POMCP gets 0.429 ± 0.183 and FV-POMCP gets 0.511 ± 0.124.

Next, we also applied POMCP to a more complex domain. This domain is an extension of the POMDP benchmark domain *RockSample*, that models Mars-rover exploration [150]. Consider a collaborative task where a human **H** wants to take samples of some rocks from Mars, with the help of a rover **R** deployed on Mars. There are some number of hours during the day (working hours) when **H** can control **R** herself but for the rest of the day, **R** has to behave autonomously. Not all types of rocks are of equal scientific value; **H** knows the value of each of these rocks but **R** does not. (Once again, we assume that **H** cannot communicate these values to **R** directly.)

Formally, consider an instance of *RockSample* on a $m \times m$ grid, with n rocks, each of which belong to one of k different types. The state space is a cross-product of the x- and y-coordinate of **R** with n binary features $IsSampled_i = \{Yes, No\}$, which indicate which of the rocks have already been sampled. (Each rock can only be sampled once.)

RockSample is a turn-based game: **R** may first take $l_{\mathbf{R}}$ steps in any of the four cardinal direction (during those hours when it is running autonomously) after which **H** may similarly take $l_{\mathbf{H}}$ steps (during the remaining hours). Thus, the set of actions available to **H** is the set of all trajectories of length exactly $l_{\mathbf{H}}$ while that available to **R** is the set of all trajectories

Algorithm 3 Adapted POMCP for CIRL games

procedure SEARCH(h)
  repeat
    if $h = empty$ **then**
      $s \sim I$
    else
      $s \sim B(h)$
    SIMULATE($s, h, 0$)
  until TIMEOUT()
  return $argmax_{a^R} V(ha^R)$

procedure ROLLOUT($s, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  $a^R, a^H \sim Uniform(A^R), Uniform(A^H)$
  $s' \sim T(s, a^R, a^H)$
  return $r(s) + \gamma \cdot$ ROLLOUT($s', ha^R a^H, depth + 1$)

procedure SIMULATE($s, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  if $h \notin T$ **then**
    return ROLLOUT($s, h, depth$)
  $a^R \leftarrow argmax_{a^R} V(ha^R) + c\sqrt{\frac{\log N(h)}{N(ha^R)}}$
  $\theta \leftarrow s_\theta$
  $a^H \leftarrow$ SAMPLEHUMANACTION(θ, h, a^R)
  $s' \sim T(s, a^R, a^H)$
  $R \leftarrow r(s) + \gamma \cdot$ SIMULATE($s', ha^R a^H, depth + 1$)
  $B(h) \leftarrow B(h) \cup \{s\}$
  $N(h) \leftarrow N(h) + 1$
  $N(ha^R) \leftarrow N(ha^R) + 1$
  $N(ha^R a^H) \leftarrow N(ha^R a^H) + 1$
  $V(ha^R) \leftarrow V(ha^R) + \frac{R - V(ha^R)}{N(ha^R)}$
  $V_\theta(ha^R a^H) \leftarrow V_\theta(ha^R a^H) + \frac{R - V_\theta(ha^R a^H)}{N_\theta(ha^R a^H)}$
  return R

procedure SAMPLEHUMANACTION(θ, h, a^R)
  $a^H \sim \pi_H\left(a^H \mid V_\theta(ha^R a^H) + c\sqrt{\frac{\log N_\theta(ha^R)}{N_\theta(ha^R a^H)}}\right)$
  return a^H

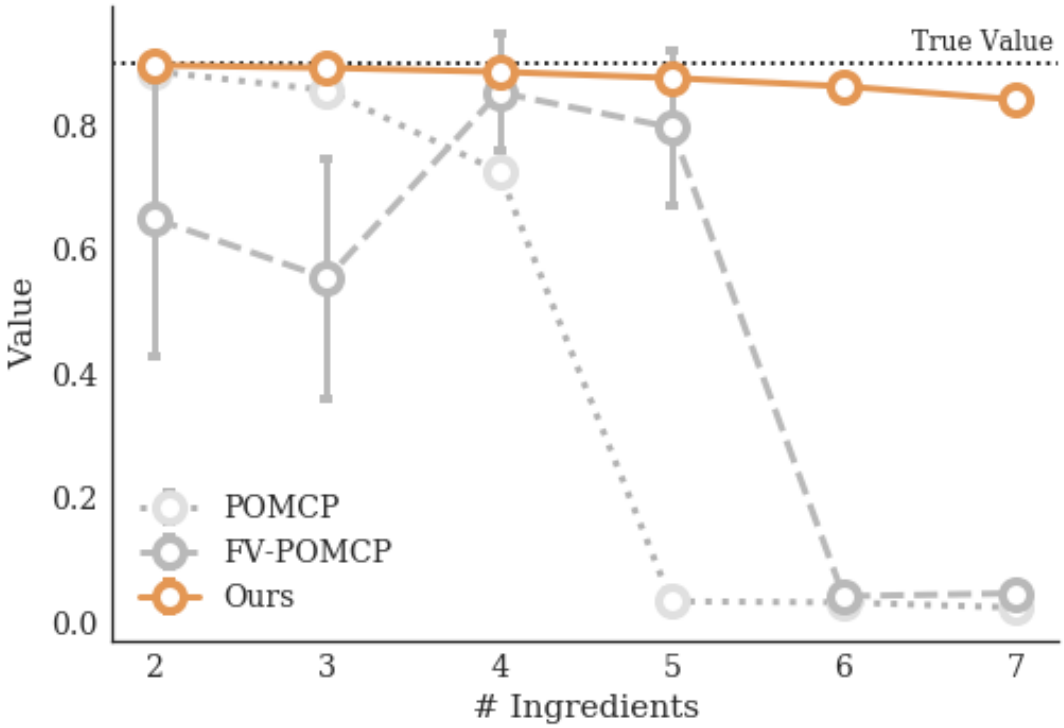


Figure 5.10: The value attained by POMCP, FV-POMCP, and our adapted algorithm in 30,000 samples with various numbers of ingredients. (Right) Value attained by POMCP, FV-POMCP and our approximate algorithm with 2 recipes and 6 ingredients.

with length at most $l_{\mathbf{R}}$. ($\mathbf{R}$ may wait on any specific step if it requires more information while $\mathbf{H}$ may not wait since they has all the information required.)

The set of all reward parameters Θ is composed of a collection of k-dimensional vectors, where the i^{th} entry represents the reward received for sampling rock i. Both agents receive the reward specified by the true reward parameter θ when they sample a rock and receive no reward for any other action.

Details of Experiment

We repeated our experiment with a 5×5 grid ($m = 5$), 3 types of rocks ($k = 4$), and 4 rocks total ($n = 4$).

This domain is much more complex than the cooking domain. For example, note that for even the simplest version of this domain, with $l_{\mathbf{H}} = l_{\mathbf{R}} = 1$, $|\mathcal{A}_{\mathbf{H}}| = 4$ and $|\mathcal{A}_{\mathbf{R}}| = 5$ (since $\mathbf{R}$

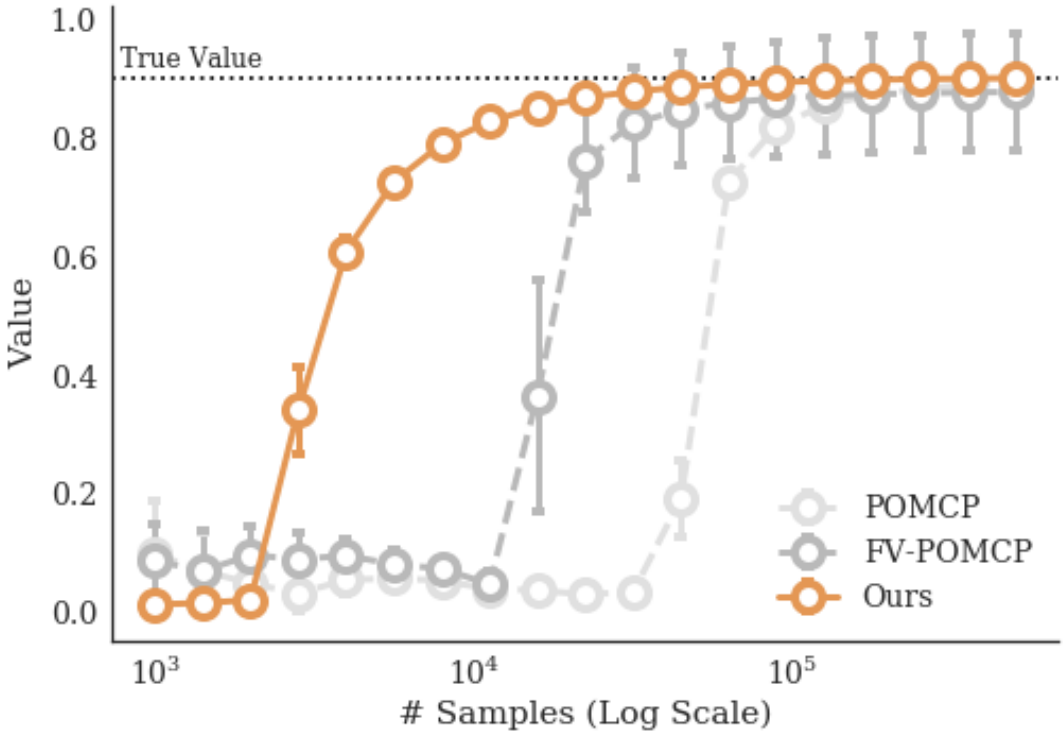


Figure 5.11: Value attained by POMCP, FV-POMCP and our approximate algorithm with 2 recipes and 6 ingredients.

can also choose to wait); if we raise $l_{\mathbf{R}}$ slightly to 2, we have $|\mathcal{A}_{\mathbf{R}}| = 13$. Hence, we only ran our experiments with POMCP, which scales the best of the three types of the algorithms.

We found similar results to those from ChefWorld. For any value of $l_{\mathbf{R}}$ beyond 1, FV-POMCP and POMCP fail to solve the problem; the branching factor of the search tree they both construct is too large and thus, both methods deplete the 16GB of memory in our system almost immediately. Our method however manages to scale to larger values of $l_{\mathbf{H}}$ and $l_{\mathbf{R}}$ with relative ease; our method successfully computed the optimal policy for this domain with values of $l_{\mathbf{H}} = l_{\mathbf{R}} = 2$ within two hours of computation.

Chapter 6

Robust Alignment

This chapter discusses the equilibrium selection problem that can arise in assistance games and the need to consider strategy robustness in the study of assistance games. The chapter begins with an application of the tools from the previous chapter to compare the performance of pedagogic solutions with individual reward maximizing solutions. As one might expect, pedagogic solutions lead to higher utility for **H**. Because even the pedagogic **H** still has to pay a lot of attention to immediate reward, **R**'s best-response still performs well even if **H** is *not* behaving pedagogically.

Next, we consider a setting that admits purely pedagogic solutions that have no connection to immediate reward. The setting is a modification of the supervision problem from Section 3.1 where **H** needs to learn which actions produce reward from experience. Because **H**'s actions only impact **R**'s information state, all actions produce the same immediate reward in an assistance game formulation.

We will show that there exists a class of solutions where **H** acts as if **R** will copy their action and approximately maximizes the expected reward of their selections given the reward observations so far. A single strategy for **R** that occasionally takes exploration actions for **H** can successfully assist *any* strategy for **H** from this family of alternatives. This represents a relatively robust alignment solution: the combined system $\mathbf{R} \circ \mathbf{H}$ is guaranteed to eventually take the optimal action with probability 1, so long as **H** satisfies relatively simple constraints on their policy.

These solutions, however, may take arbitrarily long to settle on the optimal action. As in CIRL, $\mathbf{R} \circ \mathbf{H}$ can perform better if **H** reveals more information about their goal. In this case however, the best **H** can do is reveal information about their history of reward observations. In some cases, **H** may be able to communicate enough information that **R** can behave as if it directly observes reward. This means that $\mathbf{R} \circ \mathbf{H}$ can implement the *optimal* learning strategy.

However, this performance improvement comes at the cost of robustness. Because **H** *only* cares about how **R** interprets their actions, there are multiple distinct pedagogic solutions that arise from equally informative but distinct ways to encode their reward observations. Furthermore, misalignment in these encoding schemes can lead $\mathbf{R} \circ \mathbf{H}$ to perform arbitrarily

poorly, even though **H**'s behavior reveals a lot of information about their goal. In the worst case, $\mathbf{R} \circ \mathbf{H}$ may even *minimize* utility for **H**. This is unlike the ChefWorld example, where the pedagogic solutions needed to be informative and maximize immediate reward. As a result, we consider two additional modes of interaction: a turn-taking mode where **H** and **R** alternate choosing arms, and a preemptive mode, where **R** decides to pull an arm or ask **H** to pull an arm. As one might expect, the communicative policy WSLS is now less attractive — although other, more complicated, communication strategies may do a better job of trading off between conveying information and maximizing utility. Although the need to consider both sources of utility typically increases the burden on **H** it can help manage alignment problems by making it easier for **H** and **R** to coordinate.

6.1 Exploring strategy robustness in ChefWorld

To further investigate the performance of CIRL-inspired alignment solutions in realistic settings (e.g., where **H** may not be rational and **H**'s strategy is not known perfectly in advance), we ran another experiment. We varied whether **H** behaved according to CIRL or IRL, **R**'s model of **H** in training (rational or Boltzmann-rational), and the actual model of **H** (same as previous). We measured the proportion of times the team prepared the correct recipe in each setting, fixing the number of ingredients at 3 and recipes at 4. Figure 6.1 shows the primary results.

Averaged across different models of **H** used to train **R**, when **H** behaved according to CIRL, **H** and **R** succeeded in preparing the correct recipe $> 90\%$ of the time. This was also true when **H** behaved Boltzmann-rationally. This suggests that the pedagogic behavior that arises from CIRL makes it more robust to any sub-optimality from **H**. In contrast, when **H** behaved as in IRL (i.e., not pedagogically), they only prepared the correct recipe $\sim$70% of the time when **H** was rational, and $\sim$40% of the time when **H** was not. So, the importance of pedagogic behavior from **H** to achieve value alignment is clear.

Additionally, these results suggest that the pedagogic behavior that arises from CIRL makes it more robust to sub-optimality from **H**. When **H** demonstrated sub-optimal behavior but behaved pedagogically according to CIRL, **H** and **R** were only $\sim$2% less successful at preparing the correct recipe than if **H** behaved optimally; however, when **H** demonstrated sub-optimal behavior but behaved according to IRL, they were $\sim$40% less successful.

To further investigate the properties that lead to effective alignment, and the consequences of strategy mismatch, we did a second study with 10 possible human policies instead of 2. The 10 policies were chosen from a 5×2 factorial of the human's behavior and presence of bias. The 5 possible behaviors were rational, Boltzmann-rational with $\beta = 1$, Boltzmann-rational with $\beta = 5$, ϵ-greedy with $\epsilon = 0.1$, and ϵ-greedy with $\epsilon = 0.01$. The 2 possible levels for presence of bias were "No Bias" and "Bias", where "Bias" denoted that **H** had a systematic preference for choosing the "Wait" action. (In our setting, **H** received a reward of 0.25 every time they chose the "Wait" action.)

Human's Actual Behavior		Robot's Model of Human: CIRL, Rational	CIRL, Boltzmann-Rational	IRL, Rational	IRL, Boltzmann-Rational	**Average**
CIRL	Rational	1	1	1	0.754 ± 0.43	**0.9385**
	Boltzmann-Rational	0.961 ± 0.16	0.969 ± 0.16	0.972 ± 0.17	0.752 ± 0.44	**0.9135**
IRL	Rational	0.743 ± 0.46	0.686 ± 0.47	0.698 ± 0.45	0.667 ± 0.47	**0.6985**
	Boltzmann-Rational	0.357 ± 0.47	0.409 ± 0.49	0.378 ± 0.47	0.484 ± 0.50	**0.407**
	Average	**0.76525**	**0.766**	**0.762**	**0.66425**	

Figure 6.1: The proportion of times that **H** and **R** prepared the correct recipe on the cooking domain when **R** is trained with, and **H** actually behaves according to, a variety of different behaviors. They were significantly more successful at preparing the correct recipe when **H** behaved pedagogically according to CIRL.

The results of our experiment are presented below in Figure 6.2 as a heat map. Much like the simpler experiment, we find that **H** and **R** are much more successful when **H** behaves pedagogically and that in the presence of pedagogic behavior, the team's performance is more robust to any sub-optimality from **H**.

6.2 Assisting a Learning H

Next, we turn to the problem of strategy robustness in earnest. We consider a setting where **H** is subject to additional informational constraints. Namely, we assume that **H** has to learn about θ from observations of reward over time. This changes the nature of **R**'s strategy. Instead of observing **H** in order to learn about θ, **R** also has to account for **H**'s information state. It is only after **H** makes enough observations about reward that their behavior can be effectively learned from. We illustrate the difference between assisting a learner and an expert with an example.

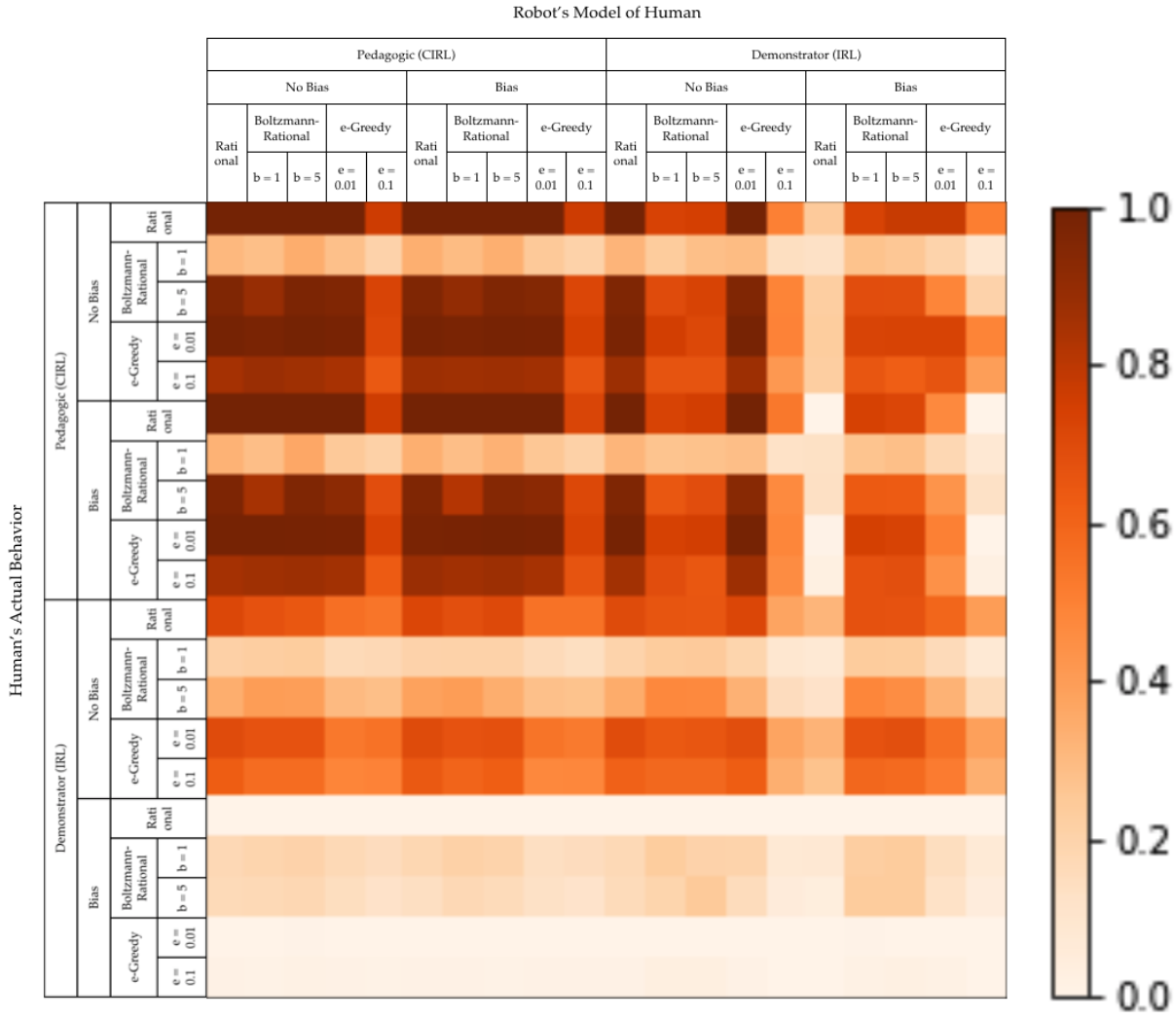


Figure 6.2: The proportion of times that **H** and **R** prepared the correct recipe on the cooking domain with 3 ingredients and 4 recipes when **R** is trained with, and **H** actually behaves according to, a variety of different behaviors. They were significantly more successful at preparing the correct recipe when **H** behaved pedagogically by following a policy specified by CIRL.

At the start of every day, the human **H** would like to consume a single caffeinated beverage, either coffee or tea. Before the human wakes up, the robot can prepare one of the two caffeinated beverages or choose not to. If the robot does not make a beverage, then the human will prepare one of the two beverages. The human has an unknown but fixed preference toward either coffee or tea, represented by $\Theta = \{0, 1\}$. The human and the robot share a uniform prior over Θ. The human gets θ reward for consuming tea and $1 - \theta$ reward for consuming coffee, which they will observe after consuming either the tea or coffee. In addition, the act of preparing coffee or tea costs the human $\varepsilon = 0.1$ points of reward, but if the tea or coffee was already prepared by the robot **R**, then no such penalty is incurred.

If the human knew the true value of θ in the above example, then she can maximize reward by preparing tea when $\theta = 1$ and coffee when $\theta = 0$. The best policy for the robot would then be wait one day to see what beverage the human chooses, then make that beverage every day after that. However, if the robot performs this policy when the human does not know the true value of θ, then this policy will lead to an expected reward of 0.5 a day, lower than the 0.9 that the human can achieve if the robot always takes no action. Instead, the optimal policy for the robot would be to prepare a drink for the human — to help the human to learn the value of θ — and then let the human prepare a drink so the human can signal the value to the robot. This policy has expected reward 1 a day for each day after the second.

Interestingly, **R** can help **H** perform better even if the cost ε of making beverages for the human is 0, if **H** is not following the optimal (reward-maximizing) policy, as the following example demonstrates. Suppose that the cost of making beverages for the human is $\varepsilon = 0$, and that the human **H** is a Q-learner with initial value -1 that follows a greedy policy for choosing tea or coffee. Thus, **H** makes tea with probability 1 if **H**'s Q-value for tea is higher, and vice versa in the case where their Q-value for coffee is higher (in the case where their Q-values are equal, they chooses one at random). No matter what beverage they sample randomly in the first timestep, **H** will always make that beverage afterwards due to a bad initialization and under-exploration. This leads to an average reward per episode of 0 with probability 0.5. However, if **R** prepares both beverages, **H**'s Q-values will update to reflect which beverage they like better. In turn, this allows **R** to observe **H**'s preference and eventually produce utility at the optimal rate.

The key observation here is that **R**'s optimal policy depends meaningfully on **H**'s information state. If **R** overestimates the information **H** can provide about rewards, then we observe persistent misalignment — in a similar, unrecoverable way to the overoptimization results in Chapter 3. Appropriately characterizing the scope of information about how to evaluate world state is one of the central challenges in designing aligned (or alignable) systems.

Remark 11. *In the design of autonomous systems, designers are faced with a tradeoff between providing the system with enough information about the goal so that* **R** *can be effective without making unrealistic assumptions about* **H***'s capabilities and knowledge state.*

Learning from a learner

In an MAB, there are K distinct actions, each with its own reward distribution. At each timestep, the actor selects an action, called an 'arm', from this set of options and observes a reward sampled from the corresponding distribution. This allows them to update their estimate of the reward distribution for that arm. The objective, as above, is to maximize the sum of discounted rewards.

Formally, we can define an MAB as a type of POMDP.

Definition 25. (Multi-Armed Bandit)
Let Θ parameterize a space of K-dimensional reward distributions $\Theta \in \Delta(\mathbb{R})^K$. Then, a multi-armed bandit M is a POMDP *with $\mathcal{S} = \Theta$, $\mathcal{A} = [1, \ldots, K]$, and static transitions $T = \delta$. Furthermore, the observation space is the set of real numbers with $P_{\mathcal{O}|\mathcal{S}}(\theta, k) = \theta_k$, where θ_k is reward distribution for arm k.*

We use $k^{(t)} \in [1, \ldots, K]$ to represent the arm selected at time t. A reward observation $u^{(t)} \sim \theta_{k^{(t)}}$ is sampled from the corresponding arm distribution. A *strategy* is a mapping that determines the correct distribution to sample from given a history of reward observations and previous arm pulls: $K_t(k^{(1)}, u^{(1)}, \ldots, k^{(t-1)}, u^{(t-1)})$.

We use μ_k to represent the mean of arm k, with parameters θ_k. We use j^* to represent the index of the best arm and μ^* to represent its mean. $T_k(t)$ represents the number of pulls of arm k up to and including time t. The goal of this game is to maximize the sum of rewards over time, or alternatively, to minimize the expectation of the regret $\bar{R}(t)$, defined as:

$$\bar{R}(t) = \sum_t (\mu^* - \mu_{k^{(t)}}) = \sum_k (\mu^* - \mu_k)\, T_k(t). \tag{6.1}$$

Before formalizing the problem of *assisting* a human who is learning, rather than noisily-optimal, we look at *passively inferring* the reward from their actions. We call this the *inverse bandit* problem.

Definition 26. *(Inverse Multi-Armed Bandit)*
Let M be a multi-armed bandit problem. Let $\pi_{\mathbf{H}}$ be a bandit strategy that maps histories of past actions and rewards to distributions over arm indices. $\pi_{\mathbf{H}} : a_{\mathbf{H}}^{(1)} \times u^{(1)} \times \cdots \times a_{\mathbf{H}}^{(t-1)} \times u^{(t-1)} \to \Delta(K)$. In the associated inverse bandit problem the goal is to recover θ by observing only the arm pulls of $\mathbf{H}$ over time $a_{\mathbf{H}}^{(1)}, ..., a_{\mathbf{H}}^{(t)}$.

Unlike the (stationary) IRL case, $\mathbf{H}$ does not have access to the true reward parameters. $\mathbf{H}$ receives the reward signal r_t sampled according to θ. As a consequence, the human arm pulls are not i.i.d.; the distribution of human arm pulls changes as they learn more about their preferences.

With this, we can look at the performance of inference (or lack thereof) when $\mathbf{R}$ has an incorrect model of $\mathbf{H}$. Specifically, we will fix a particular multi-armed bandit and compare how well $\mathbf{R}$ is able to identify θ when $\mathbf{H}$ employs different learning strategies. We will show

that learning is possible, depending on the structure of **H**'s policy, although a learning **H** does reveal less information than one that can act directly based on θ. Next, we show that inference performs poorly when **R** models **H** as an expert but they are actually learning.

In our experiments, we used a horizon 50 Beta-Bernoulli bandit with four arms. Pulling the ith arm produces a reward of one with probability θ_i and zero with probability $1 - \theta_i$: $\Theta = [0, 1]^4$. We assume a uniform prior over Θ: $\theta_i \sim \text{Beta}(1, 1)$.

We consider 5 classes of human policy:

- **ϵ-greedy**, a learning **H** that chooses the best arm in hindsight with probability $1 - \epsilon$ and a random arm with probability ϵ.[1]
- **WSLS**, the *win-stay-lose-shift* policy [126] sticks with the arm pulled in the last round if it returned 1, and otherwise switches randomly to another arm.
- **TS**, the *Thompson-sampling* policy [166] maintains a posterior over the arm parameters, and chooses each arm in proportion to the current probability it is optimal. This is implemented by sampling a particle from the posterior of each arm, then pulling the arm associated with the highest value.
- **UCL**, the *upper-credible limit* policy [125] is an algorithm similar to Bayes UCB [84] with softmax noise, used as a model of human behavior in a bandit environment.[2]
- **GI**, the *Gittins index* policy [54] is the Bayesian optimal solution to an infinite horizon discounted objective MAB.[3]

In addition, we also defined the following noisily-optimal human policy to serve as a baseline:

- **ϵ-optimal**, a *fully informed* **H** that knows the reward parameters θ, chooses the optimal arm with probability $1 - \epsilon$, and chooses a random action with probability ϵ.[4]

To investigate the miscalibration that occurs when we do not model learning behavior we use the Metropolis-Hastings algorithm [108] to approximate the posterior over reward parameters θ given human actions. We compare the posterior we get when we model learning with the posterior that assumes **H** is ϵ-optimal.

We compared the log-density of the true reward parameters in the posterior conditioned on 5 **H** actions, under both models, when **H** is actually learning. We report the results in Table 6.1. For every learning human policy, we find that the log-density of the true reward parameters is significantly higher when we model learning than when we do not. In the case

[1] We performed grid search to pick an ϵ based on empirical performance, and found that $\epsilon = 0.1$ performed best.

[2] We set $K = 4$ and softmax temperature $\tau = 4$.

[3] We follow the approximations described by Chakravorty and Mahajan in [30], and choose a discount rate ($\gamma = 0.9$) that performs best empirically using grid search.

[4] We set ϵ to match that of the ϵ-greedy policy.

Table 6.1: Log-density of true reward params in a horizon 5 inverse MAB

	Assumed **H** Policy	
Actual **H** Policy	**Correct Policy**	**ϵ-Optimal**
ϵ-greedy	0.49	-0.23
WSLS	0.95	0.13
TS	0.02	-0.23
UCL	0.03	-0.30
GI	0.94	0.20
ϵ-Optimal	–	1.55

of ϵ-greedy and TS, we find that the posterior which fails to model learning assigns negative log-density to the true parameters. This means the posterior is a *worse estimate* of θ than the prior.

It is harder to assist a learner

In the *assistive multi-armed bandit*, we have a joint system $\mathbf{R} \circ \mathbf{H}$ that aims to do well in an MAB M. This model is an extension of the supervision-POMDP considered in Section 3.1. The primary change is in the space of allowed policies for $\mathbf{H}$, which can only depend on θ through the history of reward observations.

In each round:

1. $\mathbf{H}$ selects an arm to suggest based on the history of previous arm pulls *and rewards*: $\pi_{\mathbf{H}}(a_{\mathbf{R}}^{(1)}, u^{(1)}, \ldots, a_{\mathbf{R}}^{(t-1)}, u^{(t-1)}) \in [1, \ldots, K]$.

2. $\mathbf{R}$ selects which arm to actually execute based on the history of the human's attempts and the actual arms chosen: $\pi_{\mathbf{R}}(a_{\mathbf{H}}^{(1)}, a_{\mathbf{R}}^{(1)}, \ldots, a_{\mathbf{H}}^{(t-1)}, a_{\mathbf{R}}^{(t-1)}, a_{\mathbf{H}}^{(t)}) \in [1, \ldots, K]$.

3. $\mathbf{H}$ observes the current round's arm and corresponding reward: $(a_{\mathbf{R}}^{(t)}, u^{(t)} \sim \theta\left(a_{\mathbf{R}}^{(t)}\right)$.

We start with a comparison between the case of assisting an expert (i.e., the model from Section 3.1) and assisting a learner. As one might expect, asssisting an expert in this simplified setting is quite easy in theory. We show this by proving that it is possible to infer the correct arm while making finitely many mistakes in expectation.

Theorem 22. *Suppose that* $\mathbf{H}$*'s arm pulls are i.i.d and let* f_i *be the probability* $\mathbf{H}$ *pulls arm* i*. If* $\mathbf{H}$ *is noisily optimal, that is,* $f_{j^*} > f_i$ *for all sub-optimal* i*, there exists a robot policy* $\mathbf{R}$ *that has finite expected regret for every value of* θ*:*

$$\mathbb{E}\left[\bar{R}(T)\right] \leq \sum_{i \neq j^*} \frac{\mu^* - \mu_i}{(\sqrt{f_{j^*}} - \sqrt{f_i})^2}$$

Proof. Our robot policy $\mathbf{R}$ simply pulls the most commonly pulled arm.

Let $\hat{f}_i(t) = \frac{1}{t}\sum_{k=1}^{t}[a_{\mathbf{H}}^{(t)} = i]$ be the empirical frequency of $\mathbf{H}$'s pulls of arm i up to time t. Note that $a_{\mathbf{R}}^{(t)} = i$ only if $\hat{f}_{j^*}(t) \leq \hat{f}_i(t)$. We apply a Chernoff bound to the random variable $\hat{f}_{j^*}(t) - \hat{f}_i(t)$. This gives that, for each i,

$$\Pr(\hat{f}_i(t) \leq \hat{f}_j(t)) \leq e^{-t(\sqrt{f_i}-\sqrt{f_j})^2}. \tag{6.2}$$

Summing Eq. 6.2 over t and suboptimal arms gives the result. □

This is in contrast to the standard results about regret in an MAB: for a fixed, nontrivial MAB problem M, any MAB policy has expected regret at least logarithmic in time on some choice of parameter θ [95, 105]:

$$\mathbb{E}\left[\bar{R}(T)\right] \geq \Omega(log(T)).$$

Several approaches based on Upper Confidence Bounds (UCB) have been shown to achieve this bound, implying that this bound is tight [95, 4, 27]. Nonetheless, this suggests that the problem of assisting a noisily-optimal human is significantly easier than solving a standard MAB.

The assistive MAB is at least as hard as a standard MAB. For the same sequence of arm pulls and observed rewards, the amount of information available to $\mathbf{R}$ about the true reward parameters is upper bounded by the corresponding information available in a standard MAB. From a certain perspective, actually *improving* on human performance in isolation is hopelessly difficult – $\mathbf{R}$ does not get access to the reward signal, and somehow must still assist a person who does.

Consistent assistance

We begin with the simplest success criterion from the bandit literature: consistency. Informally, consistency is the property that the player eventually pulls suboptimal arms with probability 0. This can be stated formally as the average regret going to 0 in the limit: $\lim_{t\to\infty} \bar{R}(t)/t = 0$. In an MAB, achieving consistency is relatively straightforward: any policy that is greedy in the limit with infinite exploration (GLIE) is consistent [126, 147]. In contrast, in an assistive-MAB, it is not obvious that the robot can implement such a policy when the $\mathbf{H}$ strategy is inconsistent. (You can trivially achieve consistency if $\mathbf{H}$ is consistent.) The robot observes no rewards and thus cannot directly estimate the best arm in hindsight.

However, it turns out a weak condition on the human allows the robot-human joint system to guarantee consistency:

Theorem 23. *If the human $\mathbf{H}$ implements a noisily greedy policy, that is, a policy that pulls the arm with highest sample mean strictly most often, then there exists a robot policy $\mathbf{R}$ such that $\mathbf{R} \circ \mathbf{H}$ is consistent.*

Proof. Fix a set of decaying disjoint exploration sequences E_k, one per arm, such that $\lim_{t\to\infty} \frac{1}{t}|E_k \cap \{1, ..., t\}| \to 0$ and $\lim_{t\to\infty} |E_k \cap \{1, ..., t\}| \to \infty$. In other words, each arm is pulled infinitely often, but at a decaying rate over time.

Let i_t be the arm most commonly pulled by $\mathbf{H}$ up until time t, and $\mathbf{R}$ be defined by

$$a_{\mathbf{R}}^{(t)} = \begin{cases} k & t \in E_k \\ i_t & \text{otherwise} \end{cases}.$$

Note that this implies that for suboptimal k, $\frac{1}{t}T_k(t) \to 0$ in probability as $t \to \infty$, as the sample means of all the arms converge to the true means, and the rate of exploration decays to zero. This in turn implies that $\mathbf{R} \circ \mathbf{H}$ achieves consistency. □

In other words, assistance is possible if the human picks the best actions in hindsight. This robot $\mathbf{R}$ assists the human $\mathbf{H}$ in two ways. First, it helps the human explore their preferences – $\mathbf{R} \circ \mathbf{H}$ pulls every arm infinitely often. This fixes possible under-exploration in the human. Second, it stabilizes their actions and helps ensure that $\mathbf{H}$ does not take too many suboptimal actions - eventually, $\mathbf{R} \circ \mathbf{H}$ converges to only pulling the best arm. This helps mitigate the effect of noise from the human.

Modeling learning as ϵ-optimality leads to inconsistency

We now investigate what occurs when mistakenly we model learning behavior as noisy-optimality. A simple way to make $\mathbf{R} \circ \mathbf{H}$ consistent when $\mathbf{H}$ is noisily optimal is for $\mathbf{R}$ to pull the arm most frequently pulled by $\mathbf{H}$.

Theorem 24. *If* $\mathbf{H}$ *plays a strategy that pulls the best arm most often and* $\mathbf{R}$ *plays* $\mathbf{H}$*'s most frequently pulled arm, then* $\mathbf{R} \circ \mathbf{H}$ *is consistent.*

Proof. Eventually, $\mathbf{H}$'s most frequent arm converges to the best arm with probability 1 by hypothesis. At this point, $\mathbf{R}$ will pull the best arm going forward and achieve a per-round regret of 0. □

Next we consider the impact of applying this strategy when its assumptions are incorrect, i.e., $\mathbf{H}$ is learning. For simplicity, we assume $\mathbf{H}$ is greedy and pulls the best arm given the rewards so far. We will consider a $1\frac{1}{2}$-arm bandit: a bandit with two arms, where one has a known expected value and the other is unknown. We show that pairing this suboptimal-learner with the 'most-frequent-arm' strategy leads the joint system $\mathbf{R} \circ \mathbf{H}$ to be *inconsistent*:

Theorem 25. *If* $\mathbf{H}$ *is a greedy learner and* $\mathbf{R}$ *is 'most-frequent-arm', then there exists an assistive-*MAB M *such that* $\mathbf{R} \circ \mathbf{H}$ *is inconsistent.*

Proof. (sketch) The proof consists of two steps. First, we show a variant of a classical bandit result: if $\mathbf{H}$ and $\mathbf{R}$ output the constant arm in the same round, they will for the rest of time. Second, we show that this occurs with finite probability and get a positive lower bound on the per-round regret of $\mathbf{R} \circ \mathbf{H}$. □

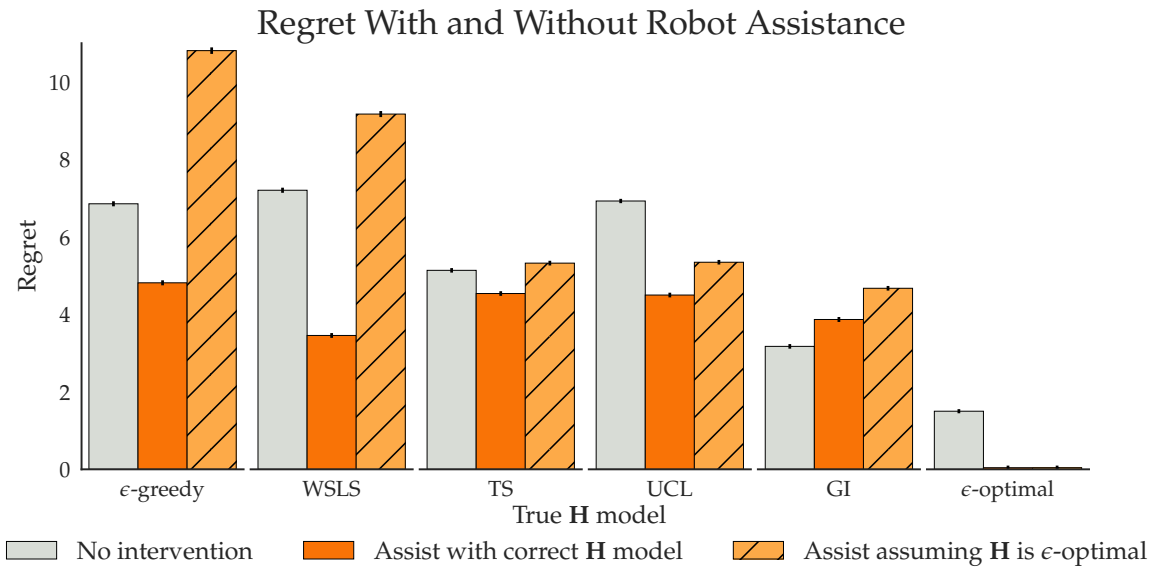


Figure 6.3: Averaged regret of various human policies (lower = better) over 100,000 trajectories when unassisted, assisted with the correct human model, and assisted assuming that the human is noisily-optimal. Assistance lowers the regret of most learning policies, but it is important to model learning: ignoring that the human is learning can lead to worse performance than no assistance. Note that assisted WSLS performs almost as well as the Gittins Index policy, an empirical verification of Theorem 26.

While this is a simplified setting, this shows that the types of mistakes and suboptimality represented by learning systems *are not* well modeled by the standard suboptimality assumptions used in research on recommendation systems, preference learning, and human-robot interaction. The suboptimality exhibited by learning systems is stateful and self-reinforcing. Figure 6.7 shows the practical impact of modeling learning. It compares an optimal assistance policy for a supervision-POMDP with an optimal policy for an assistive-MAB. In general, assistive-MAB policies seem to fit into three steps: *explore* to give **H** a good estimate of rewards; *observe* **H** to identify a good arm; and then *exploit* that information.

MABs are the standard theoretical model of reinforcement learning and so this observation highlights the point that the term inverse reinforcement learning is somewhat of a misnomer (as opposed to inverse optimal control): IRL's assumptions about an agent (noisy optimality) lead to very different inferences than *actually* assuming an agent is learning.

Comparing assistance strategies

Optimizing an assistance policy

The optimal response to a given human strategy can be computed by solving a partially observed Markov decision process (POMDP) [81]. The state is the reward parameters θ and **H**'s internal state. The observations are the human arm pulls. In this framing, a variety of approaches can be used to compute policies or plans, e.g., online Monte-Carlo planning [143, 64] or point-based value iteration [121].

In order to run experiments with large sample sizes, our primary design criterion was fast online performance. This lead us to use a direct policy optimization approach. The high per-action cost of Monte-Carlo planners makes them impractical for this problem. Further, explicitly tracking θ and **H**'s internal state is strictly harder than solving the inverse MAB.

Our approach applies the policy optimization algorithm of [40] to assistive-MABs. Given an assistive-MAB , we sample a batch of reward parameters θ from the prior $p(\Theta)$; generate trajectories of the form $\xi = [(a_{\mathbf{H}}^{(1)}, a_{\mathbf{R}}^{(1)}, u^{(1)}), ..., (a_{\mathbf{H}}^{(t)}, a_{\mathbf{R}}^{(t)}, u^{(t)})]$ from **H** and the current robot policy $\mathbf{R}^{(i)}$; and use the trajectories to update the robot policy to $\mathbf{R}^{(i+1)}$. During this offline training stage, since we are sampling reward parameters rather than using the ground truth reward parameters, we can use the generated rewards r_t to improve on $\mathbf{R}^{(i)}$.

We represent **R**'s policy as a recurrent neural network (RNN). At each timestep, it observes a tuple $(a_{\mathbf{R}}^{(t-1)}, a_{\mathbf{H}}^{(t)})$ where $a_{\mathbf{R}}^{(t-1)}$ is the most recent robot action and $a_{\mathbf{H}}$is the most recent human action. In response, it outputs a distribution over arm indices, from which an action is sampled. Given a batch of trajectories, we use an approximate policy gradient method [160] to update the weights of our RNN. In our experiments, we used Proximal Policy Optimization (PPO) [135], due to its ease of implementation, good performance, and relative insensitivity to hyperparameters.

To alleviate the problem of exploding and vanishing gradients [120], we use Gated Recurrent Units (GRU) [33] as the cells of our recurrent neural network. The output of the GRU cell is fed into a softmax function, and this output is interpreted as the distribution over actions. To reduce to variance in our policy gradient estimate, we also use a value function baseline [36] and apply Generalized Advantage Estimation (GAE) [133]. We used weight normalization [131] to speed up training. We used a batch size of 250000 timesteps or 5000 trajectories per policy iteration, and performed 100 policy iterations using PPO.

To quantitatively test the hypotheses that our learned models successfully assist, we perform a two factor ANOVA. We found a significant interaction effect, $F(3, 999990) = 1778.8$, $p < .001$, and a post-hoc analysis with Tukey HSD corrections showed that we were able to successfully assist the human in all four sub-optimal learning policies ($p < .001$).

We found that our learned assistant leads to worse performance in the Gittins Index case, $q = 23.15, p < 0.001$. This is a consequence of our choice to learn a policy via policy optimization, instead of using a perfect planner. Because it is very hard to reliably improve on a Gittins Index human's behaviour, and the variance in our policy gradient estimator is substantial, our learned policy fell short of the no-intervention baseline.

Table 6.2: Increase in Reward from Robot Assistance

	Assumed **H** Policy					
Actual **H**	**ϵ-greedy**	**WSLS**	**TS**	**UCL**	**GI**	**ϵ-optimal**
ϵ-greedy	**2.13**	-0.60	-2.18	-2.20	-0.11	-3.95
WSLS	0.94	**3.75**	0.80	0.10	-2.21	-1.97
TS	0.33	**0.66**	0.60	0.44	-1.53	-0.19
UCL	1.76	-1.19	**2.51**	2.43	0.74	1.28
GI	-1.09	**-0.28**	-0.77	-0.85	-0.71	-1.50
ϵ-optimal	0.24	1.17	1.24	1.28	-3.09	**1.46**

6.3 Reward Communication Equilibria in Assistive-MABs

In the previous experiments, we assumed knowledge of the correct learning policy. In this experiment, we consider the implications of incorrectly modeling learning. We took the policies we trained in the previous section and tested them with every human policy. We report the net change in reward in Table 6.3. We colored cases where the robot **R** successfully assists the human **H** green, and cases where it fails to assist red. We bolded the best performance in each row.

Modeling learning (even with the incorrect model) generally leads to lower regret than assuming ϵ-optimal for every learning **H** policy. However, when the robot has the wrong model of learning, it can fail to assist the human. For example, ϵ-greedy is only successfully assisted when it is correctly modeled. This argues that research into the learning strategies employed by people in practice is an important area for future research.

An intriguing result is that assuming ϵ-greedy *does* successfully assist all of the suboptimal learning policies. This suggests that, although some learning policies must be well modeled, learning to assist some models can be transferred to other models in some cases. On the other hand, trying to assist GI leads to a policy that hurts performance across the board. In future work, we plan to identify classes of learners which can be assisted by the same robot policy.

Having argued that we can achieve consistency for such a broad class of human policies in an assistive MAB, we now return to the question of achieving low regret. In particular, we investigate the conditions under which $\mathbf{R} \circ \mathbf{H}$ achieve $O(\log(T))$ expected regret, as is possible in the standard MAB. For any given human **H**, there exists a robot **R** such that $\mathbf{R} \circ \mathbf{H}$ does as well as **H**: let **R** copy the **H**'s actions without modification; that is, $a^{(t)}_{\mathbf{R}} = a^{(t)}_{\mathbf{H}}$ for all t. So in the case where **H** achieves $O(\log(T))$ regret by itself, $\mathbf{R} \circ \mathbf{H}$ can as well. However, a more interesting question is that of when we can successfully assist a suboptimal **H** that achieves $\omega(\log(T))$ regret.

Remark 12. *While one may hypothesize that better human policies lead to better perfor-*

mance when assisted, this is surprisingly not the case, as the next section demonstrates.

An inconsistent policy that is easy to assist

In looking at the table above, **R** is most helpful when **H** plays the classic bandit strategy 'win-stay-lose-shift' (WSLS) [126]. As the name suggests, WSLS sticks with the current arm if the most recent reward is one:

$$a_{\mathbf{H}}^{(t)} = \begin{cases} a_{\mathbf{R}}^{(t-1)} & u^{(t-1)} = 1 \\ \mathbf{Unif}(\{k | k \neq a_{\mathbf{R}}^{(t-1)}\}) & u^{(t-1)} = 0 \end{cases} \tag{6.3}$$

This a simple strategy that performs somewhat well empirically – although it is easy to see that it is not consistent in isolation, let alone capable of achieving $O(\log(T))$ regret. Indeed, it achieves $\Theta(T)$ regret, as it spends a fixed fraction of its time pulling suboptimal arms.

However, if we look at Figure 6.3, the assisted variant of WSLS performs on par with the unassisted *optimal* learning strategy, the Gittins index policy. It turns out that, if **H** can implement WSLS, the combined system can implement an arbitrary MAB strategy from the standard MAB setting, including those that achieve logarithmic regret. In other words, the robot can successfully assist the human in efficiently balancing exploration and exploitation *despite only having access to the reward parameter through an inconsistent human.*

Theorem 26. *If* **H** *implements the WSLS strategy in a Beta-Bernoulli assistive-*MAB*, then there exists a robot strategy* **R** *such that* $\mathbf{R} \circ \mathbf{H}$ *achieves the minimal regret from the standard* MAB *with the same parameters.*

Proof. The pair $(a_{\mathbf{R}}^{(t-1)}, a_{\mathbf{H}}^{(t)})$ directly encodes the previous reward $u^{(t-1)}$. This means that $\pi_{\mathbf{R}}$can be an arbitrary function of the history of arm pulls and rewards and so it can implement any MAB policy, including the one that achieves minimal regret. □

Figure 6.4 illustrates this result by comparing a rollout of this $\mathbf{R} \circ \mathbf{H}$ with the potentially optimal arms according to the Gittins index policy for the comparable, standard, MAB.

Mutual information bounds team performance

The WSLS policy is not unique in that it allows $\mathbf{R} \circ \mathbf{H}$ to obtain logarithmic regret. A less interesting, but similarly effective policy, is for the human to directly encode their reward observations into their actions; the human need not implement a sensible bandit policy. For example, the following purely communicative **H** also works for a Beta-Bernoulli bandit:

$$a_{\mathbf{H}}^{(t)} = \begin{cases} 0 & r_{t-1} = 0 \\ 1 & r_{t-1} = 1 \end{cases} \tag{6.4}$$

We can generalize the results regarding communicative policies using the notion of mutual information, which quantifies the amount of information obtained through observing the human arm pulls.

Let $\mathbf{I}(X;Y)$ be the mutual information between X and Y, $\mathbf{H}(X)$ be the entropy of X, and $\mathbf{H}(X|Y)$ be the entropy of X given Y.

Theorem 27. *Suppose that the probability the robot pulls a suboptimal arm at time t is bounded above by some function $f(t)$, that is $P(\mathbf{R}^{(t)} \neq j^*) \leq f(t)$. Then the mutual information $\mathbf{I}(j^*; a_{\mathbf{H}}^{(1)} \times \cdots \times a_{\mathbf{H}}^{(t)})$ between the human actions up to time t and the optimal arm must be at least $(1 - f(t)) \log K - 1$.*

Proof. We can consider the multi-armed bandit task as one of deducing the best arm from the human's actions. This allows us to apply Fano's inequality [41] to $P(\mathbf{R}^{(t)} \neq j^*)$, and using the fact that the entropy of a Bernoulli random variable is bounded above by 1, we get

$$\begin{aligned} P(\mathbf{R}^{(t)} \neq j^*) \log(K-1) &\geq \mathbf{H}(\hat{j}^*|j^*) - \log 2 \\ &= \mathbf{H}(\hat{j}^*) - \mathbf{I}(\hat{j}^*; j^*) - 1 \\ &\geq \log K - \mathbf{I}(j^*; a_{\mathbf{H}}^{(1)} \times \cdots \times a_{\mathbf{H}}^{(t)}) - 1. \end{aligned}$$

Rearranging terms and using $P(\mathbf{R}^{(t)} \neq j^*) \leq f(t)$, we get

$$\begin{aligned} \mathbf{I}(j^*; a_{\mathbf{H}}^{(1)} \times \cdots \times a_{\mathbf{H}}^{(t)}) &\geq \log K - f(t) \log(K-1) - 1 \\ &\geq (1 - f(t)) \log K - 1. \end{aligned}$$

□

Intuitively, since the probability of error is bounded by $f(t)$, in $(1-f(t))$ cases the human actions conveyed enough information for A to successfully choose the best action out of N options. This corresponds to $\log N$ bits, so there needs to be at least $(1 - f(t)) \log N$ bits of information in $\mathbf{H}$'s actions.

Corollary 27.1. *Suppose that the probability the robot pulls a suboptimal arm at time t is bounded above by some function $f(t)$, that is $P(\mathbf{R}^{(t)} \neq j^*) \leq f(t)$. Then the mutual information $\mathbf{I}(a_{\mathbf{R}}^{(1)} \times u^{(1)} \times \cdots \times a_{\mathbf{R}}^{(t-1)} \times u^{(t-1)}; a_{\mathbf{H}}^{(1)} \times \cdots \times a_{\mathbf{H}}^{(t)})$ between the human actions up to time t and the human observations must be at least $(1 - f(t)) \log K - 1$.*

Proof. Since the best arm is independent of the human actions given the human observations, this follows immediately from the data processing inequality and proposition 27. □

In order to achieve regret logarithmic in time, we must have that $P(K_t \neq j^*) \leq \frac{C}{t}$ for some $C > 0$. Applying proposition 27 above implies that we must have

$$\mathbf{I}(j^*; a_{\mathbf{H}}^{(1)} \times \cdots \times a_{\mathbf{H}}^{(t)}) \geq (1 - \frac{C}{t}) \log K - 1$$

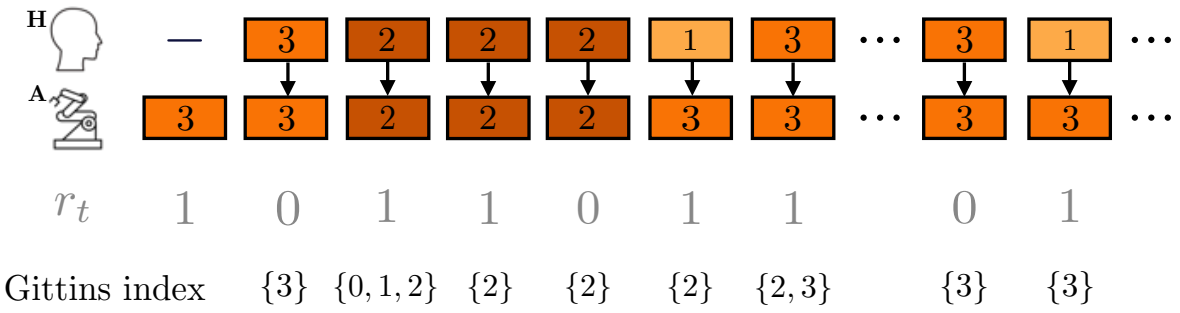


Figure 6.4: In Theorem 26 we show that it is possible to match the regret from optimal learning in an standard MAB when assisting the 'win-stay-lose-shift' (WSLS) policy. This is because WSLS perfectly communicates the observed rewards to $\mathbf{R}$. Here we show an example trajectory from an approximately optimal policy assisting WSLS. At the top is what $\mathbf{H}$ suggests, followed by what $\mathbf{R}$ pulls, followed by the reward $\mathbf{H}$ observes. For comparison, we show the arms selected by the near-optimal Gittins index policy for each belief state. This highlights the importance of communicative learning policies in an assistive-MAB.

Note that the term $\mathbf{I}(\hat{j}^*; a_{\mathbf{H}}^{(1)} \times \cdots \times a_{\mathbf{H}}^{(t)})$ depends on *both* the human policy and the robot policy - no learning human policy can achieve this bound unless the human-robot system $\mathbf{R} \circ \mathbf{H}$ samples each arm sufficiently often. *As a consequence, simple strategies such as inferring the best arm at each timestep and pulling it, cannot achieve the* $\Theta(\log T)$ *lower bound on regret.* Our WSLS results agree with Theorem 26. Assisted WSLS achieves a regret of 3.5, close to the regret of the best-performing unassisted policy, 3.2. The gap in reward is due to our choice to employ approximate policy optimization. Qualitative analysis of WSLS trajectories found that it would often switch from its current arm after only a single observation of failure, although this is clearly sub-optimal. We provide an example trajectory in Figure 6.4. The actions selected are almost identical to those of the optimal policy. This suboptimality also accounted for the small increase in regret when assisting the Gittins index policy in Figure 6.3.

Theorem 27 implies that high mutual information is required for good team performance. To verify this, we computed the mutual information for a variety of combined policies after 5 timesteps. Figure 6.5 plots this against the regret of the combined system. We consider several variants of ϵ-greedy and TS that are more or less deterministic. We consider $\epsilon \in [0, 0.02, 0.05, 0.1]$. To make TS more deterministic, we use the mean of a sample of n particles to select arms. We consider $n \in [1, 2, 3, 10, 30, \infty]$.

Across this data, higher mutual information is associated with lower assisted regret, $r(10) = -.82$, $p < .001$. Furthermore, by looking at the ϵ-greedy and TS results as a sequence, we can observe a clear and distinct pattern. Policies that are more deterministic tend to be easier to help. This is supported by the results in Table 6.1, which shows that it

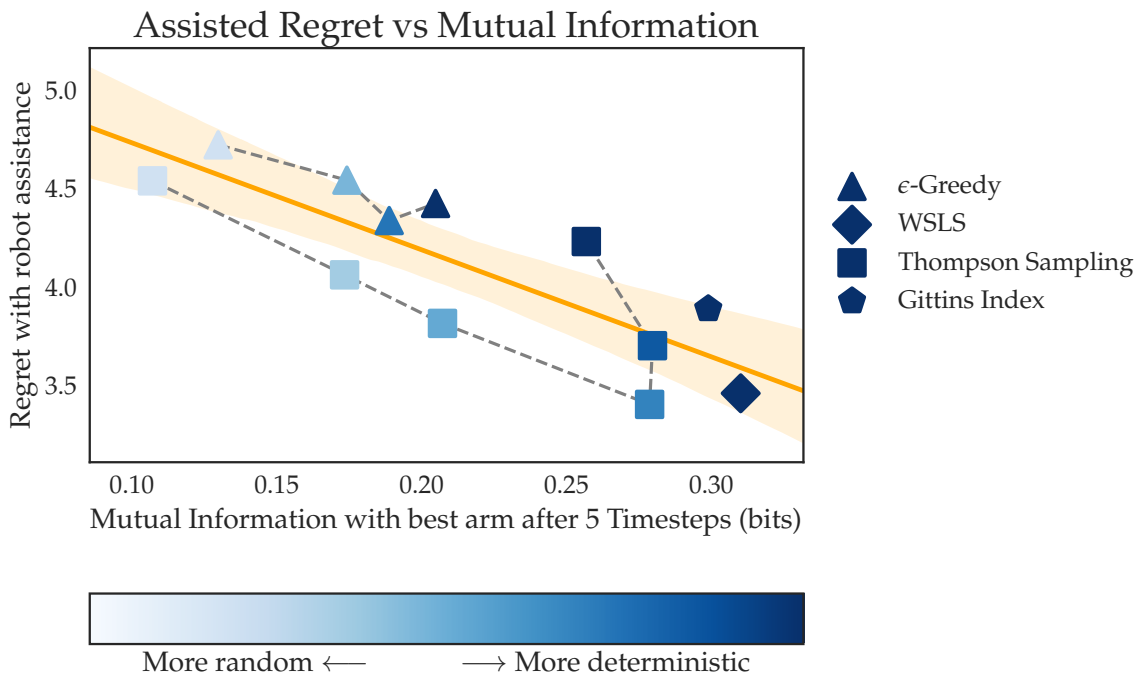


Figure 6.5: The assisted regret of various policies, plotted against the mutual information between the best arm and the policy's actions in the first 5 timesteps. We also plot the best-fit line, with 95% confidence interval, for the regression between assisted regret and mutual information. We augmented our policies with variants of ϵ-greedy and Thompson sampling with less randomness. Policies with high mutual information lead to lower regret when assisted, supporting our theoretical findings.

is easier to infer reward parameters for WSLS and GI (i.e., the two policies with the highest mutual information) than TS and ϵ-greedy.

The assistive multi-armed bandit considered so far only captures one mode of interaction. It is straightforward to consider extensions to different modes. We consider two such modes. The first is turn taking, where **H** and **R** take turns selecting arms. This can be more difficult because the robot has to act in early rounds, when it has less information, and because the human has to act in later rounds, when **H** may be noisy and the best arm has already been identified.

The second variant we consider is preemptive interaction. In this case, **R** goes first and either pulls an arm or lets **H** act. This creates an exploration-exploitation tradeoff. **R** only observes **H**'s arm pulls by actually allowing **H** to pull arms and so it must choose between observing **H**'s behavior and exploiting that knowledge.

Figure 6.6 compares the performance of different assumptions about **H** for both of these interaction modes. The results are largely similar to those of the original model. We are

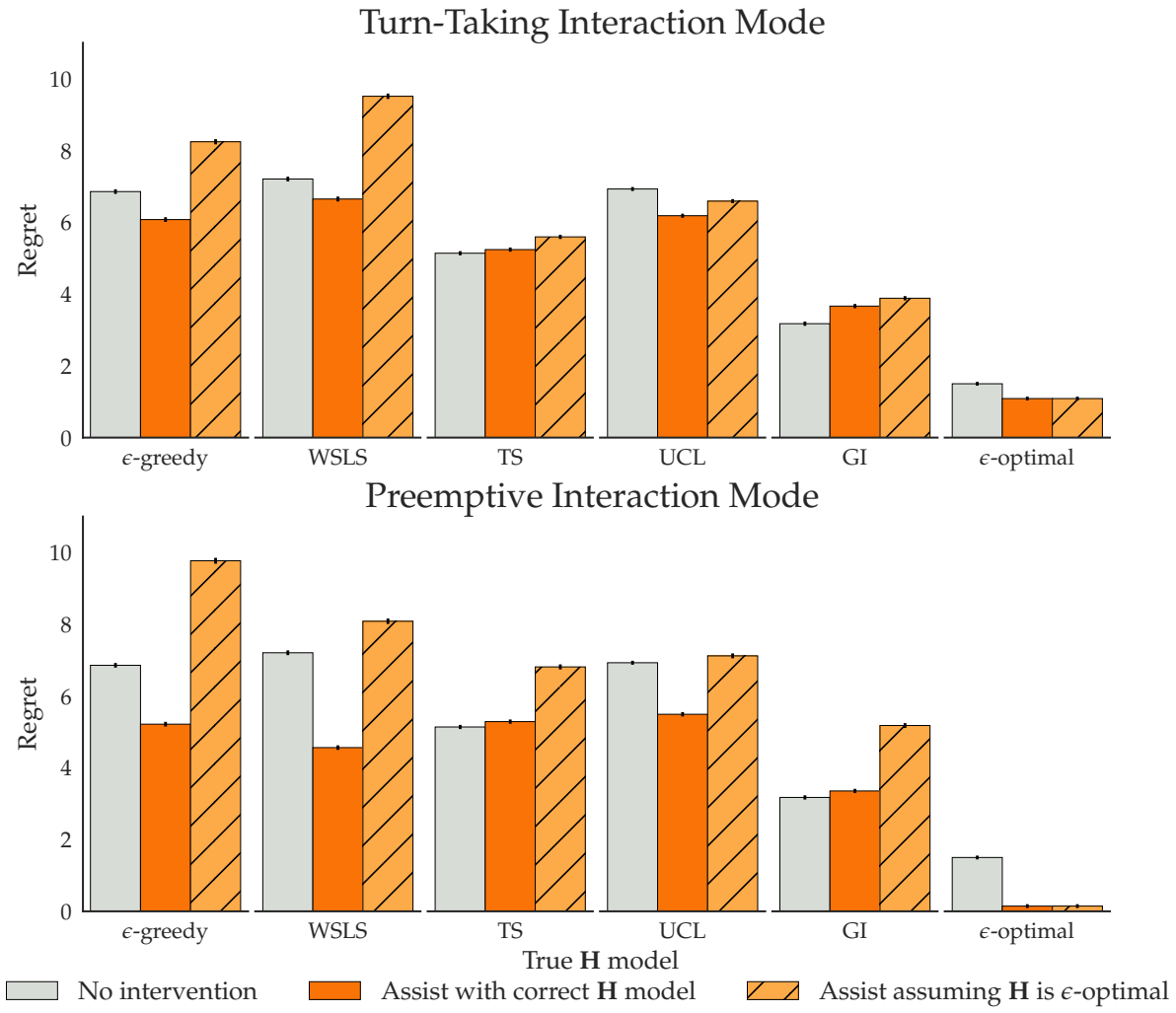


Figure 6.6: Averaged regret of various human policies (lower = better) over 100,000 trajectories under the turn-taking and preemptive interaction modes. Assistance lowers the regret of ϵ-greedy, WSLS, and UCL in both the preemptive and turn-taking interaction modes. Assistance while ignoring learning is worse than no assistance in almost every case. This offers further support for the importance of modeling learning when assisting humans.

able to assist the suboptimal policies and modeling learning as ϵ-optimality increases regret in all cases. We obtain results that confirm Theorem 27: more deterministic policies that reveal more information are easier to help.

Additionally, we see that WSLS is a less attractive policy in these settings: because **H** actions are always executed when they are observed, it no longer makes sense for **H** to employ a *purely* communicative policy. However, because actions have an effect on utility directly (i.e., **H** actually interacts with the world, rather than only providing information to **R**) this breaks the symmetry between 'win-stay-lose-shift' and 'win-shift-lose-stay.' Thus, this loss in utility is compensated for by ruling out a perverse outcome where **H** and **R** use strategies from mismatched pedagogical equilibria in a way that causes $\mathbf{R} \circ \mathbf{H}$ to minimize utility.

Explore, observe, then exploit

In looking at the policies learned for the preemptive interaction mode, we see an interesting pattern emerge. Because the policy has to choose between selecting arms directly and observing **H**, by looking at rollouts of the learned policy we can determine when it is most useful to observe **H**. We find that a clear pattern emerges. **R** initially *explores* for **H**: it selects arms uniformly to give **H** a good estimate of θ. Then, **R** *observes* **H**'s arm pulls to identify the optimal arm. For the final rounds, **R** *exploits* this information and pulls its estimate of the optimal arm. Figure 6.7 compares a representative trajectory with one that is optimized against an ϵ-optimal **H**.

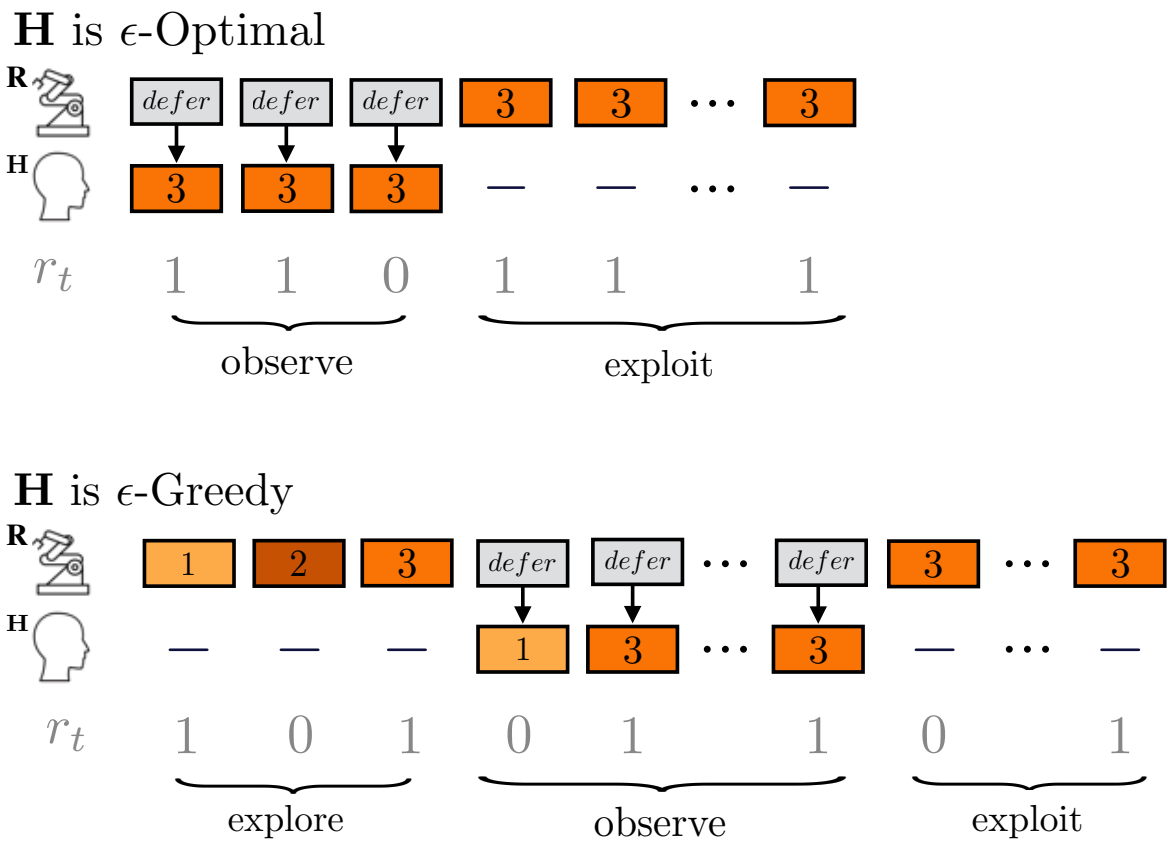


Figure 6.7: A comparison between assisting an ϵ-optimal **H** and an ϵ-greedy **H** in a modified assistive-MAB where the robot **R** has to choose between acting and letting **H** act. This creates a direct exploration-exploitation tradeoff that makes it easier to qualitatively analyze **R**'s behavior. At the top is whether **R** defers to the human or pulls an arm, followed by what **H** pulls (if the robot defers), followed by the reward **H** observes. When the robot models learning, the policy it learns has a qualitative divide into three components: explore, where the robot explores for the human; observe, when the robot lets the human pull arms; and exploit, when the robot exploits this information and pulls its estimate of the best arm. Crucially, the explore component is only found when learning is modeled. This illustrates Theorem 25, which argues that assisting an ϵ-optimal **H** is different from assisting a learning **H**.

Chapter 7

Related Work

In this chapter we collect work related to the analysis of the principal—agent value alignment problem in artificial intelligence.

7.1 The Economics of Principal–Agent Relationships and Incomplete Contracting

In Chapter 1, we discussed how the development of computational methods to select optimal actions for a given set of incentives was a crucial step in the development of AI systems. As a result, incentive alignment issues for artificial agents have clear analogues in the human principal–agent problem long studied by economists and legal scholars. Kerr [85] provided an early description of a wide variety of misaligned incentives in the aptly titled "On the folly of rewarding A, while hoping for B." Gibbons [51] provides a useful survey of principal–agent models and their applications.

Reasons for incompleteness

As discussed in Chapter 1, the principal–agent problem in economics is studied as a problem of optimal contracting. In this theory, contracts specify payments and allocation of resources to the contracting parties in different states of the world. This is analogous to the reward function or objective used to specify autonomous behavior in AI approaches. A *complete contingent* contract specifies these incentives for every possible world state that can be reached. If complete contingent contracts were possible, perfect alignment between principal and agent would be possible. However, it is understood that contracts are routinely, and likely unavoidably, incomplete. Economists have identified multiple reasons for contractual incompleteness:

- unintended incompleteness: contract designers fail to identify all circumstances that affect the value of the contract. This is sometimes referred to as bounded rationality

[145, 175].

- economizing on costly cognition and drafting: contract designers choose not to invest in the costly cognitive effort of discovering, evaluating and drafting contract terms to cover all circumstances [141, 142, 137].

- economizing on enforcement costs: contract designers leave out terms that are costly to enforce because they require more or more costly evidence or because the potential for disputes and/or court errors increases with the complexity of the contract [86, 136, 74].

- non-contractibility: some contingencies and/or actions are left out because either they cannot be observed or, even if observable, they cannot be verified by enforcers at reasonable cost. This might be because of hidden information or it might be because it is not possible to communicate (describe) the contingency or action in unambiguous terms [63, 107].

Contractual incompleteness is typically framed as undesirable. In most of the settings considered in this work, this is the case. In Section 3.2, for example, **H**'s inability to specify a complete representation of preferences led to lost utility. In addition to that, economists have also considered settings in which completeness is feasible but not optimal. These are cases in which information at the time of contracting is incomplete and new information is anticipated in the future:

- planned renegotiation: rather than writing a complete contract based on incomplete information at the start of a contracting relationship, contract designers choose to write an incomplete contract, which they expect to renegotiate in the future once more information becomes available [19, 74]

- optimal completion of contract by third-party: rather than writing a complete contract based on incomplete information at the start of a contracting relationship, contract designers choose to write an incomplete contract, which they expect to have filled in by a third-party adjudicator with better information in the future [66, 142]

In this work, we can view the dynamic incentives protocol from Section 4.3 as an example of planned renegotiation. By designing an agent to limit the scope of the initial incentives, and update them in response to the observation of a new environment, the team is able to efficiently manage the transfer of normative information (i.e., information about how to evaluate different states of the world) from **H** to **R**.

This is not a perfect analogy because there exists a complete specification of rewards which would work across all environments. The assistive-Mab introduced in Section 6.2 is a better model of a scenario where a complete contract would be suboptimal. **H** does not know how to evaluate the different options at the start of the game, and so it would be

suboptimal to specify complete incentives for which arm **R** pulls. Instead, **H** does best when they are able to update **R**'s incentives in response to reward observations made over time.

As a final category of incomplete contracts, economists, and many legal scholars, have identified strategic behavior as another reason for incompleteness:

- strategic protection of private information: a party with private information about a missing contingency does not prompt contracting to cover the contingency because doing so will reveal private information that reduces the value of the contract [155, 12]
- deterring strategic investments in costly cognition : the parties choose not to cover all contingencies because learning about them would be biased and wasteful, partly motivated by efforts to protect against strategic wealth transfers that will occur if the contingency arises [167]
- strategic ambiguity:the parties choose not to include all known and contractible contingencies in order to control strategic behavior in response to other noncontractible contingencies [17]

This class of reasons for incompleteness does not translate as readily to the principal–agent problems considered in this work. The models considered for this dissertation have focused on a dyadic relationship between a system designer and a robot. In dyadic relationships between people, strategic incentives typically arise from the fact that people, in general, have different goals. In contrast, an autonomous agent's preferences are much more of a blank slate. However, it is easy to see that these strategic concerns will arise if we consider interactions with more than one person. Consider, e.g., a model with 2 humans, a system designer and a system user. The divergence of the two human's objectives will create incentives for the human to strategically conceal information about their preferences that could be used against them in the future. To analyze these questions, future work should draw upon the implications of mechanism design [112] research as it applies to the design of complex autonomous systems. Fickinger et al. [44] describes an initial analysis in this setting and shows that, if the human players are directly rewarded for a large enough fraction of their action, it is possible to circumvent several negative results from voting theory.

Multitask models and distorted incentives

One of the central results from the study of incomplete contracting is the observation that the structure of the optimal incomplete contract should take that incompleteness into account. A well-studied example of this is the case of multitask principal–agent problems, where a agent must divide its effort across a variety of distinct tasks.

Holmstrom and Milgrom [78] and Baker, Gibbons, and Murphy [13] show that the optimal incentive contract for a task that can be measured should take into account the impact of those rewards on effort put towards tasks that cannot be measured. In particular, it may be better to reduce the quality of incentives on the measurable task below what is feasible, in

order to reduce the distortion introduced in the unmeasurable task. The model of incomplete proxy metrics from Section 3.2 can be seen as an extreme example of this, where some tasks can be measured perfectly while others can not be measured at all.

Holmstrom and Milgrom [78] gives the example of paying a teacher a fixed salary rather than one contingent on students' (easily measurable) standardized test scores. At first glance, it seems like the latter approach would be a good way to leverage incentives to improve learning outcomes. However, overly strong incentives leads to the phenomenon of "teaching to the test," where teachers dedicate too much classroom time to specific test preparation activities (e.g., process of elimination methods for multiple choice questions). This leads to less time spent on difficult-to-measure skills like creative problem solving and, in effect, the test becomes uncorrelated from these learning goals.

Baker, Gibbons, and Murphy [13] gives the example of an auto repair shop rewarding mechanics for completed repairs and thereby inducing mechanics to mislead customers about the need for repairs: completion of repairs is easily measurable; the reliability of a mechanic's diagnosis of car problems is not. More generally, sometimes it is better for a contract not to include easily contractible actions in order not to further distort incentives with respect to non-contractible actions.

When we consider the role of distorted incentives in the design of AI systems, we can observe a key difference between an artificial agent and a human one. In the absence of explicit incentives from the contract, a human will fill in the gaps with nuanced and intelligent interpretation. While this might not be desirable from the perspective of managing strategic behavior, the ability to draw on commonsense interpretations is incredibly valuable.

Contracts do not exist in a vacuum; they come heavily embedded in social and institutions structures [61]. At a minimum, they depend on shared language and organized structures for enforcement: formal enforcement through courts and coercive authorities and informal enforcement through social sanctions such as collective (coordinated) criticism and exclusion from valuable relationships. Incomplete contracts depend even more extensively on these external, third-party institutions: not only to enforce contractual terms but to supply contractual terms by interpreting ambiguous terms and filling in gaps. They contain not only their express terms but also their interpreted and implied terms [42]. This important point has been emphasized by legal scholars, in a field known as relational contracting, for several decades [99, 102, 101, 103, 55].

In the design of AI systems, it is crucial to develop methods that imitate, in general, this reliance on, and sensitivity to, external normative structure. This could include explicit mechanisms for an AI agent's objective to reference externally defined or controlled metrics or research into the cognitive processes that set these incentives for people. This concept overlaps with the economist's definition of relational contracting (see [52] for a review) to the extent that it focuses on informal sanctions for breach of obligations. Legal scholars in the sociological tradition have always included here a wide variety of sanctions (including internalized sanctions such as guilt and shame). Economists, on the other hand, typically have modeled the termination of valuable economic relationships or the degradation of reputation, which reduces contracting opportunities with third-parties in the future. The legal

concept of relational contracts goes further than enforcement to focus also on the importation of obligations into a contractual relationships from sources other than the express language of an agreement. Williamson [175], Alchian and Demsetz [6] and Klein, Crawford, and Alchian [87] were the first economic treatments to focus on this aspect of the legal analysis of relational contracts.

7.2 Impact Minimization

In Section 3.3, we identify two potential strategies to avoid the negative results about guaranteed overoptimization of proxy metrics: impact minimization methods and dynamic incentives protocols. While dynamic incentives have received less academic attention, several papers have considered generic proposals for impact measurement and penalization.

Armstrong and Levinstein [11] proposes the inclusion of a large impact penalty in the AI agent's utility function. In this work, they suggest impact be measured as the divergence between a distribution of states of the world and the distribution if the AI agent had not existed. Thus, distributions sufficiently different would be avoided. Alternatively, other approaches use an impact regularizer learned from demonstration instead of one explicitly given [9].

Krakovna et al. [92] expand on this work by comparing the performance of various implementations of impact minimization in an AI Safety Gridworld suite [97]. They propose a reachability condition which penalizes the system for removing the ability to navigate the system to states that were previously reachable. In doing so, the hope is that the system will avoid unrecoverable failures.

Turner, Hadfield-Menell, and Tadepalli [169] considers an analogous method where the agent is penalized for changes in a vector of value functions, where the value functions are computed based on a set of candidate reward functions. This forces the agent to accomplish its goal with minimal effect on the state of alternative tasks. Turner, Ratzlaff, and Tadepalli [168] shows that this method is effective at avoiding side-effects in an adaptation of Conway's 'Game of Life' [3].

Krakovna et al. [91] provides a theoretical grounding of these approaches in a setting where the agent must preserve the ability to perform future tasks. They formalize the problem of *interference,* where the consideration of future tasks leads the agent to perform tasks that are useful for the overall distribution of tasks and show how to modify the objective to remove this incentive. This approach is quite interesting in the context of the combined theoretical solution discussed at the end of Chapter 3: it defines a regularizer that explicitly prevents $\mathbf{R}$ from interfering with the ability to maximize future (proxy) objectives. Thus, we can see this method, combined with the dynamic incentives protocol in Section 4.3 as an interesting proposal for a value alignment solution based on proxy objectives.

7.3 Inverse Reinforcement Learning

From a technical standpoint, the starting point for the analysis in this thesis resides in the framework of inverse reinforcement learning. Ng and Russell [115] define inverse reinforcement learning (IRL) as follows:

> **Given:** 1) measurements of an agent's behavior over time, in a variety of circumstances, 2) if needed, measurements of the sensory inputs to that agent; 3) if available, a model of the environment.
> **Determine** the reward function being optimized.

In Section 2.3, we give an formulation of the problem. Here, we go over some of the key results in the area.

Ng and Russell [115] use a mathematical programming approach to characterize the set of potential IRL solutions for a given set of demonstrations. They identify one of the central challenges in IRL: for any demonstration, there are typically many reward functions that could be optimal. In the context of this dissertation, this essentially states that a demonstration of an optimal trajectory is an incomplete representation of **H**'s goal.

Abbeel and Ng [2] considered the problem of IRL with linear reward functions and showed that a policy that matches the feature expectations from the demonstrations will always get the same amount of reward as the expert. Ratliff, Bagnell, and Zinkevich [123] show how to use subgradients to efficiently compute weights that maximize the margin between the demonstrations and alternative trajectories. Ramachadran and Amir [122] and Ziebart et al. [181] both considered the problem of learning reward functions from noisy behavior. Ramachadran and Amir [122] use Markov-chain Monte Carlo methods to compute an approximate posterior over reward functions. The key idea in their approach is to combine a specific proposal distribution of candidate rewards with a method for efficiently re-using the solutions to previous reward functions.

Ziebart et al. [181] highlights the importance of modeling the details of the choice set considered by the demonstrator. They show the difference between a distribution of behavior that selects actions in proportion to the Q function and one that selects trajectories in accordance with their total utility. While this is structurally similar, the difference has to do with the way that noise accumulates. As an example, they consider an MDP with 3 trajectories, all of equal utility. They show that the policy which selects actions in proportion to their Q-value leads to an uneven distribution over trajectories cause by the environment dynamics. They introduce a dynamic programming method to compute this distribution on trajectories and use it to propose an effective gradient-based method for IRL.

Syed and Schapire [162] describes an adversarial approach to IRL and apprenticeship learning. They consider a setting where an adversary can select a reward function in order to maximize the regret of a deployed policy. Interestingly, their method allows imitation to improve on demonstrator performance, depending on the structure of the deployment domain.

A recent area of focus in IRL research is the development of methods that can scale to nonlinear reward spaces like neural networks. Finn, Levine, and Abbeel [45] proposed a method that uses policy learning instead of planning to deal with high-dimensional problems. Ho and Ermon [77] proposed an adversarial approach that uses a neural network trained to distinguish optimized and demonstrated trajectories to train an imitation policy. Fu, Luo, and Levine [48] learn state-only rewards with this approach and show that their method can learn reward functions that are disentangled from the problem dynamics in the sense that they create incentives the optimal policy across a set of transition dynamics.

Natarajan et al. [113] introduce an extension to IRL where R observes multiple actors that cooperate to maximize a common reward function. This is a different type of cooperation than we consider, as the reward function is common knowledge and R is a passive observer. Waugh, Ziebart, and Bagnell [172] and Kuleshov and Schrijvers [93] consider the problem of inferring payoffs from observed behavior in a general (i.e., non-cooperative) game given observed behavior. Combining these methods with the assistance game formulations considered in this work is an interesting direction in which to study the problem of assisting groups of humans with competing objectives.

7.4 Reward Learning

In addition to IRL there are a variety of methods for reward learning that have been studied in artificial intelligence. There is a vast number of preference learning methods for non-sequential problems [49]. Methods that learn reward functions from preferences, surveyed in Wirth, Fürnkranz, and Neumann [176], are particularly relevant to our work.

Christiano et al. [34] learn a reward function from preferences over pairs of trajectories, by sampling trajectories from a learned policy and querying the user about pairs with high uncertainty. A similar setup is used in Wirth, Fürnkranz, and Neumann [176] and Akrour, Schoenauer, and Sebag [5], based around other policy optimization methods. It is also possible to learn reward functions from preferences on actions [49] and states [128]. Many of these methods select from already encountered trajectories, which will be guesses at optimal training behavior. Sadigh et al. [130] identify queries by gradient-based optimization in the trajectory space (in continuous environments).

Reward functions are also frequently learned from direct reward values [88, 171] and expert ratings [35]. An interesting genre of this work looks at learning from trajectory rankings, where an expert provides a total ordering over a set of possible comparisons. Brown et al. [24] give a promising approach that combines trajectory ranking to learn a distribution over reward functions that can be effectively maximized in a risk-averse fashion in novel environments. This is similar to the approach taken in IRD (where a proxy objective provides a ranking over all trajectories in the development environment) but Brown et al. [24] are able to scale their method to much larger problems and more complex reward spaces. Their approach is notable for its simplicity: rankings are converted into pairwise preferences and used to train a classifier.

Shah et al. [138] argue that these reward learning approaches can be united by observing that each models **H**'s choice as a reward-driven choice over different option sets. They catalog a wide variety of distinct reward feedback mechanisms and their associated probabilistic assumptions. In the context of this dissertation, these correspond to different choices of the action spaces for **H** $\mathcal{A}_{\mathbf{H}}$ and/or different choices of **H**'s policy $\pi_{\mathbf{H}}$.

Shah et al. [139] takes an interesting approach to reward learning that relies on the initial state of the world. The core idea is that the initial state that a robot is deployed into is rarely a truly 'random' draw from a distribution. Instead, it is often heavily optimized towards different (human) objectives. The key idea in this paper is that by considering the initial state as an optimized environment, it is possible to learn about the objectives that state was optimized towards. Crucially, this justifies the deployment of low-impact policies. In a general state of nature, there is no reason to minimize impact.

Shaikh and Goodrich [140] proposes and evaluates several interfaces for reward specification, eliciting three-way trade-offs from users. This is complementary to IRD, and could be adapted to collect data for our method. Furthermore, the divide-and-conquer strategy suggests a natural way to scale this approach to collect arbitrary n-way trade-offs because we can combine observations from several three-way interactions.

7.5 Optimal Reward Design

Singh et al. [148] formalize and study the problem of designing optimal rewards. They consider a designer faced with a distribution of environments, a class of reward functions to give to an agent, and a fitness function. They observe that, in the case of bounded agents, it may be optimal to select a proxy reward that is distinct from the fitness function. Sorg, Singh, and Lewis [154] and subsequent work has studied the computational problem of selecting an optimal proxy reward.

In Section 4.2, we considered an alternative situation where the system designer is the bounded agent. In this case, the proxy reward function is distinct from the fitness function – the true utility function in our terminology – because system designers can make mistakes. IRD formalizes the problem of determining a true utility function given an observed proxy reward function. This enables us to design agents that are robust to misspecifications in their reward function.

MacGlashan et al. [100] presents a system to ground natural language commands to reward functions that capture a desired task. Using natural language as an interface for specifying rewards is complementary to our approach as well.

7.6 Pragmatics

The pragmatic interpretation of language is the interpretation of a phrase or utterance in the context of alternatives [62]. For example, the utterance "some of the apples are red" is

often interpreted to mean that "not all of the apples are red" although this is not literally implied. This is because, in context, we typically assume that a speaker who meant to say "all the apples are red" would simply say so. Recent models of pragmatic language interpretation use two levels of Bayesian reasoning [47, 60]. At the lowest level, there is a literal listener that interprets language according to a shared literal definition of words or utterances. Then, a speaker selects words in order to convey a particular meaning to the literal listener. To model pragmatic inference, we consider the probable meaning of a given utterance from this speaker. We can think of inverse reward design from Chapter 4 as a model of pragmatic reward interpretation: the speaker in pragmatic interpretation of language is directly analogous to the reward designer in CIRL.

7.7 Corrigible Systems

Omohundro [117] identifies the instrumental goals of artificial agents: goals which are likely to be adopted as subgoals of most objectives. He identifies an incentive for self-preservation as one of these instrumental goals. Soares et al. [151] takes an initial step at formalizing these arguments. They refer to agents that allow themselves to be switched off as *corrigible* agents. They show that one way to create corrigible agents is to make them indifferent to being switched off, but in their model this also makes **R** equally likely to commit suicide as to take any other action. The key difference in our formulation in Section 4.1 is that R knows that it does not know the utility function $\mathcal{U}$. This gives a natural way to create incentives to be corrigible and to analyze the behavior if R is incorrigible.

Orseau and Armstrong [119] consider the impact of human interference on the learning process. The key to their approach is that they model the off-switch for their agent as an interruption that forces the agent to change its policy. They show that this modification, along with some constraints on how often interruptions occur, allows off-policy methods to learn the optimal policy for the given reward function just as if there had been no interference. Their results are complementary to ours. In Section 4.1, we determine situations where the optimal policy allows the human to turn the agent off, while they analyze conditions under which turning the agent off does not interfere with learning the optimal policy. We have shown that **R**'s obedience depends on a tradeoff between **R**'s uncertainty about and **H**'s rationality. However, they considered **R**'s uncertainty in the abstract. In practice **R** would need to learn about utility evaluations through **H**'s behavior. Our work analyzes how the way **R** learns about reward impacts its performance and obedience.

7.8 Intent Inference For Assistance

Instead of just being blindly obedient, an autonomous system can infer **H**'s intention and actively assist **H** in achieving it. We formalize this problem as a supervision-POMDP in Section 3.1. 'Do What I Mean' software packages interpret the intent behind what a pro-

grammer wrote to automatically correct programming errors [164]. When a user uses a telepointer network lag can cause jitter in their cursor's path. Gutwin, Dyck, and Burkitt [65] address this by displaying a prediction of the user's desired path, rather than the actual cursor path.

Similarly, in assistive teleoperation, the robot does not directly execute **H**'s (potentially noisy) input. It instead acts based on an inference of H's intent. In Dragan and Srinivasa [39] **R** acts according to an arbitration between **H**'s policy and **R**'s prediction of **H**'s policy. Like our work, Javdani, Bagnell, and Srinivasa [80] formalize assistive teleoperation as a POMDP in which **H**'s goals are unknown, and try to optimize an inference of **H**'s goal.

7.9 Cooperative Agents

Fern et al. [43] consider a hidden-goal MDP, a special case of a POMDP where the goal is an unobserved part of the state. This is similar to the assistance-POMDP model introduced in Chapter 4, where the set of possible objectives are binary functions of the goal state. The frameworks share the idea that **R** helps **H** accomplish a partially observed objective. The key difference between this model and the CIRL model considered in Chapter 5 lies in the treatment of the human (the agent in their terminology). In CIRL, we treat **H** as an actor in a decision problem that both actors collectively solve. This allows us to analyze the problem for robustness with respect to the human policy and is crucial to modeling the human's incentive to teach.

Woodward, Finn, and Hausman [177] considers the problem of training assistive agent as a meta-learning problem. They propose a policy learning approach that is able to identify effective teaching and learning policies for **H** and **R**. The primary extension beyond CIRL methods is that they consider a setting with partial observability for both **H** and **R**.

Carroll et al. [28] considers the problem of learning teamwork behaviors via self-play. Their work identifies the substantial coordination challenges that arise from multiple parallel equilibria. Unlike in adversarial settings, where an opponent that performs suboptimally typically helps the performance of a strategy, it will typically hurt a cooperative behavior. They show, via a user study, evidence of pedagogical behavior from their subjects that can counteract **R**'s mistake over time and **H** gets a better model of $\pi_{\mathbf{R}}$. This indicates the importance of studying adaptive solutions to cooperative decision problems.

Boutilier [22] introduces the problem of *multiagent*-MDPs, which formalizes these coordination problems. Boutilier proposes a simple solution that augments an MDP with a coordination state in situations where there are multiple potentially optimal actions. This allows agents to learn to coordinate online via a simple, randomized coordination mechanism. Wu et al. [178] propose a method that uses a hierarchical model to infer subgoals that agent's have selected online to allow for a more scalable form of this adaptive planning. This class of problem has also been studied in the context of *ad-hoc teamwork*, which proposes methods for evaluating the performance of agents in a setting where coordination is not possible beforehand.

7.10 Optimal Teaching

Because CIRL creates incentives for the human to teach, as opposed to maximizing reward in isolation, our work is related to optimal teaching: finding examples that optimally train a learner [14, 57, 56]. The key difference is that efficient learning is the objective of optimal teaching, while it emerges as a property of pedagogical equilibrium in CIRL.

Cakmak and Lopes [25] consider an application of optimal teaching where the goal is to teach the learner the reward function for an MDP. The teacher gets to pick initial states from which an expert executes the reward-maximizing trajectory. The learner uses IRL to infer the reward function, and the teacher picks initial states to minimize the learner's uncertainty. In CIRL, this approach can be characterized as an approximate algorithm for $\mathbf{H}$ that greedily minimizes the entropy of $\mathbf{R}$'s belief.

Beyond teaching, several models focus on taking actions that convey some underlying state, not necessarily a reward function. Examples include finding a motion that best communicates an agent's intention [38], or finding a natural language utterance that best communicates a particular grounding [58]. All of these approaches model the observer's inference process and compute actions (motion or speech) that maximize the probability an observer infers the correct hypothesis or goal. Our approximate solution to pedagogical demonstration in Section 5.1 is analogous to these approaches, in that we compute actions that are informative of the correct reward function.

7.11 Algorithms for Planning with Partial Observability

Algorithms for Mixed-Observability Decision Problems

The supervision-POMDP and assistance-POMDP formulations we consider in this work are examples of *mixed-observability* Markov decision process (MOMDP) since the state space can be factored into a fully and a partially-observable component. This structure allows for more efficient solution methods. Ong et al. [118] leverage the factored nature of the state space to create a lower dimensional representation of belief space. This core idea is orthogonal to ours, which exploits CIRL's information asymmetry instead. The two can be leveraged together.

Partially-observable Markov decision processes

We chose to explicate our modified Bellman update from Section 5.2 in the context of PBVI and POMCP because they are the seminal point-based and MCTS algorithms respectively, for solving POMDPs. For example, SARSOP [94] and DESPOT [152], two state-of-the-art algorithm for POMDPs, are variants of PBVI and POMCP respectively. The principles we outlined in Section 5.2 and can be easily adapted to a large variety of point-based and MCTS

algorithms, including any which may be developed in the future, to derive even more efficient algorithms for solving CIRL games.

Decentralized POMDPs

Dec-POMDP algorithms can be used to compute pedagogical equilibria for CIRL games directly, without using the POMDP reduction. Oliehoek and Amato [116] provides a useful introduction to the topic and an overview of solution methods. While exact solution methods are generally intractable for any but the smallest problems, recent work has made progress on this front. Such Dec-POMDP algorithms which attempt to prune away unreasonable strategies resemble our approach. Amato, Dibangoye, and Zilberstein [7] use reachability analysis to identify reachable states, then consider all policies which are useful at such states.

Hansen, Bernstein, and Zilberstein [75] model other agents possible strategies as part of a players belief, and prune away weakly dominated strategies at each step. While such approaches use heuristics to prune away some suboptimal strategies, we leverage the information structure of CIRL to compute the optimal strategy for H and prune away all other strategies. This guarantees an exponential reduction in complexity while preserving optimality; this is not true for the other methods.

Chapter 8

Directions for Future Work

My long-term research goal is to ensure that AI systems are effective in carrying out the goals of their users and designers, despite fundamental limitations on our ability to specify those goals. This dissertation introduces a simple formulation for the assistance game and developed theory and algorithms therein. This chapter describes a research agenda that builds on these results with a focus on scaling algorithms to work in complex robotics environments, developing a theory of reward learning to focus on minimizing misalignment, and extending the theory of value alignment to the case of multiple principals or value systems.

Accounting for more complex models of human (ir)rationality

The work in this dissertation has focused on principal–agent settings where the human player is modelled as behaving optimally or approximately optimally. An important next step in this research program is to consider the impact of more realistic cognitive or informational limits on the human player's strategy. Section 6.2 describes initial work in this direction: we consider the impact of a human player that is learning about her objectives over time. We discuss what it means to meaningfully assist such an agent and identify broad conditions in which assistance is possible. In the future, I intend to investigate the impact of similar restrictions to account for limited memory, planning ability, or information about the world. Additionally, it is important to consider the impact of changing preferences, individual inconsistency, and variations in values across different people and groups. This will allow us to understand the theoretical limits of robot assistance and identify minimal criteria for the principal's knowledge of and ability to communicate their objectives so that value alignment is well-defined and feasible.

An integrated design environment for behavior

We currently rely on an AI system designer to directly specify loss functions, environments, and policy space in general purpose programming languages. This requires a huge amount of knowledge and often produces brittle learning systems. Developments such as AutoML

simplify the design process and reduce the design effort for AI systems. However, they also make the system *more* reliant on the loss function selected and data provided, placing an increased burden on the designer to supervise and verify performance. To ensure that these systems perform reliably at scale, we need to design specification methods that 1) can scale to the complexity of modern AI datasets and problems; 2) can scale to the complexity of human preferences; and 3) naturally integrate oversight and verification. This research program essentially reduces the problem of designing AI systems to a human-computer interaction problem where optimization and reward learning compile the high-level language of preferences into executable behaviors. Designing AI systems in this fashion, combined with progress on other robotics challenges such as uncertainty in perceptual systems and motor control, allows us to optimize for more complex decisions, naturally integrate oversight and supervision, due to the tight relationship between oversight and training, and use learning to inform the design process and reduce overall designer effort.

Loss functions for reward learning

The analysis of the off-switch game and inverse reward design in Chapter 4 shows that AI systems should model (or at least adapt to) uncertainty in reward learning, in addition to more traditional types of uncertainty about state, dynamics, and controls. The next next question is, what properties of the robot's reward model lead to better performance? There are clear ways that learning a reward function *that will be optimized later* is a different learning problem from standard supervised ranking or classification. In future research, I will develop a theory of learned reward functions that can explain this difference. For example, perhaps there is an appropriate regularizer for ranking functions that reduces overall risk for planning. Another area to explore is learning objectives that naturally encode the risk aversion implemented with the Bayesian approach of inverse reward design. Similarly, an asymmetric loss function that penalizes predicting rewards that are too high more than rewards that are too low could help with the stability of optimized policies for learned rewards. The goal of this research is to understand the problem of learning incentives from a theoretical perspective and design reward learning methods that are naturally risk-averse and scale up to larger problems than state-of-the-art Bayesian inference can handle.

Value alignment with multiple principals.

In addition to continued work on the algorithmic and theoretical foundations of single-human single-robot value alignment, the research described in this dissertation can be extended to consider value alignment in the context of multiple principals: multiple people with different values and goals on whose behalf a system is supposed to take action. This poses a host of interesting algorithmic questions (e.g., scaling up to a large number of people), theoretical questions (e.g., what are the limits of computational social choice theory), and ethical questions (e.g., how should we handle conflicts of values within AI systems; what are the obligations of system designers to design systems that optimize for their users' benefit).

Solving a multiple-principal value-alignment problem can also involve modeling the interactions between a regulator, a company, and its AI agent. We can consider the optimal way to provide incentives for companies to, e.g., invest in oversight and safety research.

AI regulation

In the limit of many principals, the value-alignment problem is closely related to the regulation of AI systems. It formalizes the problem of designing systems that implement the values of collections of humans (contracting parties, organizations, cities, states, and so on). An example of this type of work is the development of procedures for human oversight of AI decision-making systems. Work on explainable AI attempts to provide human-understandable accounts of why a complex and opaque AI system behaves as it does. The analysis in this dissertation calls for focusing instead on what I call *justifiable* AI: providing reasons for machine decisions that are acceptable to the people they impact. While this does not directly solve the problem of balancing competing goals between people, it will allow AI systems to be subject to the normative infrastructure that exists to regulate decisions made by people. In ongoing work with colleagues, our goal is to develop a procedure that ensures that organizations (corporations, health care systems, governments, etc.) that deploy AI-based decision making systems can verify that their systems are both a) sufficiently predictable by humans and b) only make decisions that can be adequately justified by a specified principal. This creates the conditions needed to hold a person meaningfully responsible for an AI system's behavior. Thus, aligning an AI system with that principal will create incentives for the system to abide by existing regulations and laws.

Interactive embodied value alignment for a home assistant

A crucial and ubiquitous principal–agent relationship is the relationship between a home robot and its user or users. This challenge integrates the problem of scale (coordinating a diverse set of actions in a large state space over a long horizon), exploration/exploitation tradeoffs in online preference learning, and issues of value conflict among humans (e.g., disagreement over the thermostat). Furthermore, it will spur the development of methods that can account for these complexities in conjunction with issues around long-horizon planning, noisy sensors, and high-dimensional motor control. A value-alignment approach to the design of such a system allows the system to learn from multi-modal data about preferences and naturally allows for the management of uncertainty about those preferences. Such research draws on hierarchical planning [68, 32, 72], planning under uncertainty [71] and manipulation planning [69, 96].

Value alignment for content recommendation

The focus in this dissertation on value alignment is guided by a commitment to ensure that AI is developed in ways that are safe and beneficial for society. A specific research applica-

tion is the theory, design, and regulation of value-aligned content recommendation systems. These heavily optimized systems are widely deployed and are showing significant negative externalities such as addictive behaviors in young people, the spread of disinformation and conspiracy theories, and undermining the integrity of elections and social cohesion. In addition to being a major challenge for responsible AI development, devising ways to achieve better value alignment in these systems provides an important test bed to explore value alignment at scale with multiple stake-holders. The questions here include, for example, how to overcome the major challenge of scale. (Facebook, for example, confronts the impossible problem of moderating billions of posts in multiple languages on a weekly basis.) What is the correct way to monitor these systems? What is the role of the system in detecting places where oversight is needed? What methods and tools will allow us to design systems that more closely reflect the way in which humans (ethically) recommend content to each other? What limits the ability of AI systems to mediate these complex, consequential, and value-laden interactions?

Bibliography

[1] Pieter Abbeel, Adam Coates, and Andrew Y. Ng. "Autonomous helicopter aerobatics through apprenticeship learning". In: *The International Journal of Robotics Research* 29.13 (2010), pp. 1608–1639.

[2] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the 21st International Conference on Machine Learning*. 2004.

[3] Andrew Adamatzky. *Game of Life Cellular Automata*. Vol. 1. Springer, 2010.

[4] Rajeev Agrawal. "Sample mean based index policies by o (log n) regret for the multi-armed bandit problem". In: *Advances in Applied Probability* 27.4 (1995), pp. 1054–1078.

[5] Riad Akrour, Marc Schoenauer, and Michèle Sebag. "April: Active preference learning-based reinforcement learning". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2012, pp. 116–131.

[6] Armen A. Alchian and Harold Demsetz. "Production, Information Costs, and Economic Organization". In: *The American Economic Review* 62.5 (1972), pp. 777–795.

[7] Christopher Amato, Jilles Steeve Dibangoye, and Shlomo Zilberstein. "Incremental policy generation for finite-horizon DEC-POMDPs". In: *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling*. 2009.

[8] Dario Amodei and Jack Clark. *Faulty Reward Functions in the Wild*. `https://blog.openai.com/faulty-reward-functions/`. Blog. 2016.

[9] Dario Amodei et al. *Concrete problems in AI safety*. 2016. URL: `arXiv:1606.06565`.

[10] Cecilie Schou Andreassen. "Online Social Network Site Addiction: A Comprehensive Review". In: *Current Addiction Reports* 2.2 (2015), pp. 175–184.

[11] Stuart Armstrong and Benjamin Levinstein. *Low impact artificial intelligences*. 2017. URL: `arXiv:1705.10720`.

[12] Ian Ayres and Robert Gertner. "Filling Gaps in Incomplete Contracts: An Economic Theory of Default Rules". In: *The Yale Law Journal* 99 (1 1989), pp. 87–130.

[13] George Baker, Robert Gibbons, and Kevin J. Murphy. "Subjective Performance Measures in Optimal Incentive Contracts". In: *The Quarterly Journal of Economics* 109.4 (1994), pp. 1125–1156.

[14] Frank J Balbach and Thomas Zeugmann. "Recent developments in algorithmic teaching". In: *Language and Automata Theory and Applications*. Springer, 2009, pp. 1–18.

[15] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and machine learning*. 2019. URL: `https://fairmlbook.org/`.

[16] Yochai Benkler, Robert Faris, and Hal Roberts. *Network propaganda: Manipulation, disinformation, and radicalization in American politics*. Oxford University Press, 2018.

[17] B. Douglas Bernheim and Michael D. Whinston. "Incomplete Contracts and Strategic Ambiguity". In: *The American Economic Review* 88.4 (1998), pp. 902–932.

[18] Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. "The complexity of decentralized control of Markov decision processes". In: *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc. 2000, pp. 32–37.

[19] Patrick Bolton and Antoine Faure-Grimaud. "Satisficing Contracts". In: *The Review of Economic Studies* 77 (2010), pp. 937–971.

[20] Nick Bostrom. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, 2014.

[21] Neal E. Boudette. *Tesla Says Autopilot Makes Its Cars Safer. Crash Victims Say It Kills*. `https://www.nytimes.com/2021/07/05/business/tesla-autopilot-lawsuits-safety.html`. 2021.

[22] Craig Boutilier. "Sequential optimality and coordination in multiagent systems". In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. Vol. 1. 1999, pp. 478–485.

[23] Steve Brooks et al. *Handbook of Markov Chain Monte Carlo*. CRC Press, 2011.

[24] Daniel Brown et al. "Safe imitation learning via fast bayesian reward inference from preferences". In: *Proceedings of the International Conference on Machine Learning*. PMLR. 2020, pp. 1165–1177.

[25] Maya Cakmak and Manuel Lopes. "Algorithmic and human teaching of sequential decision tasks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2012.

[26] Donald T. Campbell. "Assessing the impact of planned social change". In: *Evaluation and Program Planning* 2.1 (1979), pp. 67–90.

[27] Olivier Cappé et al. "Kullback–Leibler upper confidence bounds for optimal sequential allocation". In: *The Annals of Statistics* 41.3 (2013), pp. 1516–1541.

[28] Micah Carroll et al. "On the utility of learning about humans for human-AI coordination". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 5174–5185.

[29] Pablo Castells, Neil J. Hurley, and Saul Vargas. "Novelty and Diversity in Recommender Systems". In: *Recommender Systems Handbook*. Springer, 2015, pp. 881–918.

[30] Jhelum Chakravorty and Aditya Mahajan. "Multi-Armed Bandits, Gittins Index, and its Calculation". In: *Methods and Applications of Statistics in Clinical Trials: Planning, Analysis, and Inferential Methods, Volume 2* (2014), pp. 416–435.

[31] Lawrence Chan et al. "The Assistive Multi-Armed Bandit". In: *Proceedings of the 14th ACM/IEEE International Conference on Human-Robot Interaction*. 2019, pp. 354–363.

[32] Rohan Chitnis et al. "Guided search for task and motion plans using learned heuristics". In: *Proceedings of the International Conference on Robotics and Automation*. IEEE. 2016, pp. 447–454.

[33] Kyunghyun Cho et al. *On the properties of neural machine translation: Encoder-decoder approaches*. 2014. URL: arXiv%20preprint%20arXiv:1409.1259.

[34] Paul F. Christiano et al. "Deep reinforcement learning from human preferences". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4302–4310.

[35] Christian Daniel et al. "Active Reward Learning." In: *Proceedings of Robotics: Science and Systems*. 2014.

[36] Thomas Degris, Patrick M. Pilarski, and Richard S. Sutton. "Model-free reinforcement learning with continuous action in practice". In: *Proceedings of the American Control Conference*. IEEE. 2012, pp. 2177–2182.

[37] Persi Diaconis and David Freedman. "On the consistency of Bayes estimates". In: *The Annals of Statistics* (1986), pp. 1–26.

[38] Anca D. Dragan, Kenton C.T. Lee, and Siddhartha S. Srinivasa. "Legibility and Predictability of Robot Motion". In: *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction*. 2013, pp. 301–308.

[39] Anca D. Dragan and Siddhartha S. Srinivasa. *Formalizing Assistive Teleoperation*. MIT Press, July, 2012.

[40] Yan Duan et al. *RL2: Fast Reinforcement Learning via Slow Reinforcement Learning*. 2016. URL: arXiv%20preprint%20arXiv:1611.02779.

[41] Robert Mario Fano. "Fano inequality". In: *Scholarpedia* 3.10 (2008), p. 6648.

[42] Allan E. Farnsworth. "Disputes Over Omission in Contracts". In: *Columbia Law Review* 68 (5 1968), pp. 860–891.

[43] Alan Fern et al. "A decision-theoretic model of assistance". In: *Journal of Artificial Intelligence Research* 50.1 (2014), pp. 71–104.

[44] Arnaud Fickinger et al. *Multi-Principal Assistance Games: Definition and Collegial Mechanisms*. 2020. URL: arXiv:2012.14536.

[45] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided cost learning: Deep inverse optimal control via policy optimization". In: *Proceedings of the International Conference on Machine Learning*. PMLR. 2016, pp. 49–58.

[46] Jaime F. Fisac et al. "Pragmatic-Pedagogic Value Alignment". In: *International Symposium on Robotics Research* (2017).

[47] Michael C. Frank et al. "Informative Communication in Word Production and Word Learning". In: *Proceedings of the 31st Annual Conference of the Cognitive Science Society*. 2009, pp. 1228–1233.

[48] Justin Fu, Katie Luo, and Sergey Levine. "Learning Robust Rewards with Adverserial Inverse Reinforcement Learning". In: *Proceedings of the International Conference on Learning Representations*. 2018.

[49] Johannes Fürnkranz and Eyke Hüllermeier. "Preference learning". In: *Encyclopedia of Machine Learning*. Springer, 2011, pp. 789–795.

[50] Jason Gauci et al. *Horizon: Facebook's Open Source Applied Reinforcement Learning Platform*. 2019. URL: arxiv:1811.00260.

[51] Robert Gibbons. *Incentives in organizations*. National Bureau of Economic Research Working Papers. 1998.

[52] Ricard Gil and Giorgio Zanarone. "Formal and Informal Contracting: Theory and Evidence". In: *Annual Review of Law and Social Science* 13 (2017), pp. 141–159.

[53] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[54] John C. Gittins. "Bandit processes and dynamic allocation indices". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1979), pp. 148–177.

[55] Charles J. Goetz and Robert E. Scott. "Principles of Relational Contracts". In: *Virginia Law Review* 67.6 (1981), pp. 1089–1150.

[56] Sally A. Goldman and Michael J. Kearns. "On the complexity of teaching". In: *Journal of Computer and System Sciences* 50.1 (1995), pp. 20–31.

[57] Sally A. Goldman, Ronald L. Rivest, and Robert E. Schapire. "Learning binary relations and total orders". In: *SIAM Journal on Computing* 22.5 (1993), pp. 1006–1034.

[58] Dave Golland, Percy Liang, and Dan Klein. "A game-theoretic approach to generating spatial descriptions". In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2010, pp. 410–419.

[59] C.A.E. Goodhart. "Problems of Monetary Management: The UK Experience". In: *Monetary Theory and Practice: The UK Experience*. Ed. by C.A.E. Goodhart. London: Macmillan, 1984, pp. 91–121.

[60] Noah D. Goodman and Daniel Lassiter. "Probabilistic Semantics and Pragmatics: Uncertainty in Language and Thought". In: *Handbook of Contemporary Semantic Theory. Wiley-Blackwell* 2 (2014).

[61] Mark Granovetter. "Economic Action and Social Structure: The Problem of Embeddedness". In: *American Journal of Sociology* 91.3 (1985), pp. 481–510.

[62] H. Paul Grice. "Logic and Conversation". In: *Syntax and Semantics, Volume 3: Speech Acts*. Ed. by Peter Cole and Jerry Morgan. Academic Press, 1975, pp. 43–58.

[63] Sanford Jay Grossman and Oliver D. Hart. "The Costs and Benefits of Ownership: A Theory of Vertical and Lateral Integration". In: *Journal of Political Economy* 94.4 (1986), pp. 691–719.

[64] Arthur Guez, David Silver, and Peter Dayan. "Efficient Bayes-adaptive reinforcement learning using sample-based search". In: *Advances in Neural Information Processing Systems*. 2012, pp. 1025–1033.

[65] Carl Gutwin, Jeff Dyck, and Jennifer Burkitt. "Using cursor prediction to smooth telepointer jitter". In: *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*. 2003, pp. 294–301.

[66] Gillian K. Hadfield. "Judicial Competence and the Interpretation of Incomplete Contracts". In: *Journal of Legal Studies* 23 (1994), pp. 159–184.

[67] Dylan Hadfield-Menell and Gillian K. Hadfield. "Incomplete contracting and AI alignment". In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 2019, pp. 417–422.

[68] Dylan Hadfield-Menell, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. "Optimization in the now: Dynamic peephole optimization for hierarchical planning". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 4560–4567.

[69] Dylan Hadfield-Menell et al. "Beyond lowest-warping cost action selection in trajectory transfer". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2015, pp. 3231–3238.

[70] Dylan Hadfield-Menell et al. "Cooperative Inverse Reinforcement Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016, pp. 3909–3917.

[71] Dylan Hadfield-Menell et al. "Modular task and motion planning in belief space". In: *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4991–4998.

[72] Dylan Hadfield-Menell et al. "Sequential quadratic programming for task plan optimization". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2016, pp. 5040–5047.

[73] Dylan Hadfield-Menell et al. "The off-switch game". In: *Proceedings of the International Joint Conference on Artificial Intelligence*. 2017.

[74] Maija Halonen-Akatwijuka and Oliver D. Hart. *More is Less: Why Parties May Deliberately Write Incomplete Contracts*. National Bureau of Economic Research Working Papers. 2013.

[75] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. "Dynamic programming for partially observable stochastic games". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 4. 2004, pp. 709–715.

[76] Md. Rajibul Hasan, Ashish Kumar Jha, and Yi Liu. "Excessive use of online video streaming services: Impact of recommender system use, psychological factors, and motives". In: *Computers in Human Behavior* 80 (2018), pp. 220–228.

[77] Jonathan Ho and Stefano Ermon. "Generative adversarial imitation learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 4565–4573.

[78] Bengt Holmstrom and Paul Milgrom. "Multitask principal-agent analyses: Incentive contracts, asset ownership, and job design". In: *Journal of Law, Economics, & Organization* 7 (1991), pp. 24–52.

[79] Jeff Horwitz and Deepa Seetharaman. *Facebook Executives Shut Down Efforts to Make the Site Less Divisive*. `https://www.wsj.com/articles/facebook-knows-it-encourages-division-top-executives-nixed-solutions-11590507499`. 2020.

[80] Shervin Javdani, J. Andrew Bagnell, and Siddhartha S. Srinivasa. "Shared Autonomy via Hindsight Optimization". In: *Proceedings of Robotics: Science and Systems XI*. 2015.

[81] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial intelligence* 101.1-2 (1998), pp. 99–134.

[82] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. "Reinforcement Learning: A Survey". In: *Journal of Artificial Intelligence Research* 4 (1996), pp. 237–285.

[83] Rudolf Emil Kalman. "When is a linear control system optimal?" In: *Journal of Basic Engineering* 86.1 (1964), pp. 51–60.

[84] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. "On Bayesian upper confidence bounds for bandit problems". In: *Artificial Intelligence and Statistics*. 2012, pp. 592–600.

[85] Steven Kerr. "On the folly of rewarding A, while hoping for B". In: *Academy of Management Journal* 18.4 (1975), pp. 769–783.

[86] Benjamin Klein. "Transaction Cost Determinants of 'Unfair' Contractual Arrangements". In: *The American Economic Review* 70 (2 1980), pp. 356–362.

[87] Benjamin Klein, Robert G. Crawford, and Armen A. Alchian. "Vertical Integration, Appropriable Rents, and the Competitive Contracting Process". In: *Journal of Law and Economics* 21.2 (1978), pp. 297–326.

[88] W.Bradley Knox and Peter Stone. "Interactively shaping agents via human reinforcement: The TAMER framework". In: *Proceedings of the Fifth International Conference on Knowledge Capture*. ACM, 2009, pp. 9–16.

[89] Levente Kocsis and Csaba Szepesvári. "Bandit Based Monte-Carlo Planning". In: *Proceedings of the 17th European Conference on Machine Learning*. Springer-Verlag, 2006.

[90] Victoria Krakovna. *Specification gaming examples in AI*. 2018. URL: `https://vkrakovna.wordpress.com/2018/04/02/specification-gaming-examples-in-ai/`.

[91] Victoria Krakovna et al. *Avoiding side effects by considering future tasks*. 2020. URL: `arXiv:2010.07877`.

[92] Victoria Krakovna et al. *Penalizing Side Effects using Stepwise Relative Reachability*. 2018. URL: `arXiv:1806.01186`.

[93] Volodymyr Kuleshov and Okke Schrijvers. "Inverse Game Theory". In: *Web and Internet Economics* (2015).

[94] Hanna Kurniawati, David Hsu, and Wee Sun Lee. "SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces." In: *Proceedings of Robotics: Science and Systems*. 2008.

[95] Tze Leung Lai and Herbert Robbins. "Asymptotically efficient adaptive allocation rules". In: *Advances in Applied Mathematics* 6.1 (1985), pp. 4–22.

[96] Alex X. Lee et al. "Unifying scene registration and trajectory optimization for learning from demonstrations with application to manipulation of deformable objects". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 4402–4407.

[97] Jan Leike et al. *AI safety gridworlds*. 2017. URL: `arXiv:1711.09883`.

[98] Tyler Lu, Dávid Pál, and Martin Pál. "Contextual multi-armed bandits". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 485–492.

[99] Stewart Macaulay. "Non-Contractual Relations in Business: A Preliminary Study". In: *American Sociological Review* 28.1 (1963), pp. 55–67.

[100] James MacGlashan et al. "Grounding English Commands to Reward Functions". In: *Proceedings of Robotics: Science and Systems XI*. 2015.

[101] Ian R. Macneil. "Contracts: Adjustment of Long-term Economic Relations Under Classical, Neoclassical, and Relational Contract Law". In: *Northwestern University Law Review* 72 (1978), pp. 854–905.

[102] Ian R. Macneil. "The Many Futures of Contracts". In: *Southern California Law Review* 1973-1974 (1974), pp. 691–816.

[103] Ian R. Macneil. "Values in contract: internal and external". In: *Northwestern University Law Review* 1983-1984 (1983), pp. 340–418.

[104] Dhruv Malik et al. "An Efficient, Generalized Bellman Update For Cooperative Inverse Reinforcement Learning". In: *Proceedings of the International Conference on Machine Learning*. 2018, pp. 3391–3399.

[105] Shie Mannor and John N. Tsitsiklis. "The sample complexity of exploration in the multi-armed bandit problem". In: *Journal of Machine Learning Research* 5.Jun (2004), pp. 623–648.

[106] Harry M. Markowitz. *Portfolio Selection: Efficient Diversification of Investments*. Yale University Press, 1959.

[107] Eric Maskin and Jean Tirole. "Unforeseen Contingencies and Incomplete Contracts". In: *The Review of Economic Studies* 66.1 (1999), pp. 83–114.

[108] Nicholas Metropolis et al. "Equation of state calculations by fast computing machines". In: *The Journal of Chemical Physics* 21.6 (1953), pp. 1087–1092.

[109] Smitha Milli et al. "Should Robots be Obedient?" In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. 2017.

[110] Matej Moravčík et al. "Deepstack: Expert-level artificial intelligence in heads-up no-limit poker". In: *Science* 356.6337 (2017), pp. 508–513.

[111] Iain Murray, Zoubin Ghahramani, and David MacKay. "MCMC for Doubly-Intractable Distributions". In: *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*. 2006.

[112] Roger B. Myerson. "Perspectives on Mechanism Design in Economic Theory". In: *American Economic Review* 98.3 (2008), pp. 586–603.

[113] Sriraam Natarajan et al. "Multi-agent inverse reinforcement learning". In: *Proceedings of the Ninth International Conference on Machine Learning and Applications*. IEEE. 2010, pp. 395–400.

[114] Ashutosh Nayyar, Aditya Mahajan, and Demosthenis Teneketzis. "Decentralized stochastic control with partial history sharing: A common information approach". In: *IEEE Transactions on Automatic Control* 58.7 (2013), pp. 1644–1658.

[115] Andrew Y. Ng and Stuart J. Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ACM. 2000, pp. 663–670.

[116] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

[117] Stephen M. Omohundro. "The Basic AI Drives". In: *Proceedings of the First Conference on Artificial General Intelligence*. 2008.

[118] Sylvie C.W. Ong et al. "Planning under uncertainty for robotic tasks with mixed observability". In: *International Journal of Robotics Research* 29.8 (2010), pp. 1053–1068.

[119] Laurent Orseau and Stuart Armstrong. "Safely Interruptible Agents". In: *Uncertainty in Artificial Intelligence*. 2016.

[120] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *International Conference on Machine Learning*. 2013, pp. 1310–1318.

[121] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. "Point-based value iteration: An anytime algorithm for POMDPs". In: *Proceedings of the 33rd Annual International Joint Conference on Artificial Intelligence*. Vol. 3. 2003, pp. 1025–1032.

[122] Deepak Ramachadran and Eyal Amir. "Bayesian inverse reinforcement learning". In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. 2007.

[123] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. "Maximum Margin Planning". In: *Proceedings of the 23rd International Conference on Machine Learning*. 2006.

[124] Ellis Ratner, Dylan Hadfield-Menell, and Anca D. Dragan. "Simplifying Reward Design through Divide-and-Conquer". In: *Proceedings of Robotics: Science and Systems*. 2018.

[125] Paul B. Reverdy, Vaibhav Srivastava, and Naomi Ehrich Leonard. "Modeling human decision making in generalized Gaussian multiarmed bandits". In: *Proceedings of the IEEE* 102.4 (2014), pp. 544–571.

[126] Herbert Robbins. "Some aspects of the sequential design of experiments". In: *Bulletin of the American Mathematical Society*. 58.5 (1952), pp. 527–535.

[127] R. Tyrrell Rockafellar and Stanislav Uryasev. "Optimization of conditional value-at-risk". In: *Journal of Risk* 2 (2000), pp. 21–42.

[128] Thomas Philip Runarsson and Simon M. Lucas. "Preference learning for move prediction and evaluation function approximation in othello". In: *IEEE Transactions on Computational Intelligence and AI in Games*. 2014, pp. 300–313.

[129] Stuart J. Russell and Peter Norvig. *AI: A Modern Approach*. Pearson, 2020.

[130] Dorsa Sadigh et al. "Active preference-based learning of reward functions". In: *Proceedings of Robotics: Science and Systems*. 2017.

[131] Tim Salimans and Diederik P. Kingma. "Weight normalization: A simple reparameterization to accelerate training of deep neural networks". In: *Advances in Neural Information Processing Systems*. 2016, pp. 901–909.

[132] Matthias Scheutz and Charles Crowell. "The Burden of Embodied Autonomy: Some Reflections on the Social and Ethical Implications of Autonomous Robots". In: *Workshop on Roboethics at the International Conference on Robotics and Automation, Rome*. 2007.

[133] John Schulman et al. *High-dimensional continuous control using generalized advantage estimation*. 2015. URL: `arXiv:1506.02438`.

[134] John Schulman et al. "Motion Planning with Sequential Convex Optimization and Convex Collision Checking". In: *International Journal of Robot Research* 33.9 (2014).

[135] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. URL: `arXiv:1707.06347`.

[136] Alan Schwartz and Robert E. Scott. "Contract Theory and the Limits of Contract Law". In: *The Yale Law Review* 113 (2003), pp. 541–619.

[137] Robert E. Scott and George Triantis. "Anticipating Litigation in Contract Design". In: *Yale Law Journal* 115 (2006), p. 814.

[138] Rohin Shah et al. "Benefits of Assistance over Reward Learning". In: *Proceedings of the NeurIPS 2020 Workshop on Cooperative AI* (2020).

[139] Rohin Shah et al. "Preferences Implicit in the State of the World". In: *International Conference on Learning Representations*. 2019.

[140] Meher T. Shaikh and Michael A. Goodrich. "Design and Evaluation of Adverb Palette: A GUI for Selecting Tradeoffs in Multi-objective Optimization Problems". In: *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. 2017.

[141] Steven Shavell. "Damage Measures for Breach of Contract". In: *The Bell Journal of Economics* 11.2 (1980), pp. 466–490.

[142] Steven Shavell. "On the Writing and Interpretation of Contracts". In: *Journal of Law, Economics and Organization* 22 (2006), pp. 289–311.

[143] David Silver and Joel Veness. "Monte-Carlo planning in large POMDPs". In: *Advances in Neural Information Processing Systems*. 2010, pp. 2164–2172.

[144] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–489.

[145] Herbert A. Simon. "A behavioral model of rational choice". In: *The Quarterly Journal of Economics* 69.1 (1955), pp. 99–118.

[146] Satinder Singh, Andrew Barto, and Richard Lewis. "Where Do Rewards Come From?" In: *Proceedings of the Cognitive Science Society* (31 2009).

[147] Satinder Singh et al. "Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms". In: *Machine Learning* 38.3 (Mar. 2000), pp. 287–308.

[148] Satinder Singh et al. "Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective". In: *IEEE Transactions on Autonomous Mental Development* 2 (2 2010), pp. 70–82.

[149] Richard D. Smallwood and Edward J. Sondik. "The optimal control of partially observable Markov processes over a finite horizon". In: *Operations Research* 21.5 (1973), pp. 1071–1088.

[150] Trey Smith and Reid Simmons. "Heuristic search value iteration for POMDPS". In: *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2004, pp. 520–527.

[151] Nate Soares et al. "Corrigibility". In: *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

[152] Adhiraj Somani et al. "DESPOT: Online POMDP Planning with Regularization". In: *Journal of Artificial Intelligence Research* 58 (2017), pp. 231–266.

[153] Edward J. Sondik. "The Optimal Control of Partially Observable Markov Processes". PhD thesis. Stanford University, 1971.

[154] Jonathan Sorg, Satinder P. Singh, and Richard L. Lewis. "Internal rewards mitigate agent boundedness". In: *Proceedings of the Twenty-Seventh International Conference on Machine Learning*. 2010, pp. 1007–1014.

[155] Kathryn E. Spier. "Incomplete Contracts and Signalling". In: *The RAND Journal of Economics* 23 (3 1992), pp. 432–443.

[156] Christian Stöcker. "How Facebook and Google Accidentally Created a Perfect Ecosystem for Targeted Disinformation". In: *Disinformation in Open Online Media*. Ed. by Christian Grimme et al. Springer, 2019, pp. 129–149.

[157] Marilyn Strathern. "'Improving ratings': audit in the British University system". In: *European Review* 5.3 (1997), pp. 305–321.

[158] Jonathan Stray et al. *What are you optimizing for? Aligning recommender systems with human values*. 2021. URL: `arXiv:2107.10939`.

[159] Mikael Sunnåker et al. "Approximate Bayesian Computation". In: *PLoS Computational Biology* 9.1 (2013), e1002803.

[160] Richard S. Sutton et al. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in Neural Information Processing Systems*. 2000, pp. 1057–1063.

[161] Umar Syed, Michael Bowling, and Robert E. Schapire. "Apprenticeship Learning Using Linear Programming". In: *Proceedings of the 25th International Conference on Machine Learning*. ACM. 2008, pp. 1032–1039.

[162] Umar Syed and Robert E. Schapire. "A Game-Theoretic Approach to Apprenticeship Learning". In: *Proceedings of the Twentieth Conference on Neural Information Processing Systems*. 2007, pp. 1449–1456.

[163] Aviv Tamar et al. "Policy gradient for coherent risk measures". In: *Advances in Neural Information Processing Systems*. 2015, pp. 1468–1476.

[164] Warren Teitelman. "Toward a Programming Laboratory". In: *Proceedings of the First International Joint Conference on Artificial Intelligence*. 1969, pp. 1–8.

[165] Rachel Thomas and David Uminsky. *The problem with metrics is a fundamental problem for AI*. 2020. URL: `arXiv:2002.08512`.

[166] William R. Thompson. "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples". In: *Biometrika* 25.3/4 (1933), pp. 285–294.

[167] Jean Tirole. "Cognition and incomplete contracts". In: *The American Economic Review* 99.1 (2009), pp. 265–294.

[168] Alex Turner, Neale Ratzlaff, and Prasad Tadepalli. "Avoiding Side Effects in Complex Environments". In: *Advances in Neural Information Processing Systems* 33 (2020).

[169] Alexander Matt Turner, Dylan Hadfield-Menell, and Prasad Tadepalli. "Conservative agency via attainable utility preservation". In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, pp. 385–391.

[170] Kurt Wagner. "Inside Twitter's ambitious plan to change the way we tweet". In: *Vox* (2019). URL: `https://www.vox.com/2019/3/8/18245536/exclusive-twitter-healthy-conversations-dunking-research-product-incentives`.

[171] Garrett Warnell et al. "Deep TAMER: Interactive Agent Shaping in High-Dimensional State Spaces". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. 2018, pp. 1545–1553.

[172] Kevin Waugh, Brian D. Ziebart, and J. Andrew Bagnell. "Computational Rationalization: The Inverse Equilibrium Problem". In: *Proceedings of the International Conference on Machine Learning*. 2011.

[173] Daniel Weld and Oren Etzioni. *The First Law of Robotics (a call to arms)*. Tech. rep. SS-94-03. AAAI, 1994, pp. 1042–1047.

[174] Mark Wilhelm et al. "Practical diversified recommendations on YouTube with determinantal point processes". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018, pp. 2165–2173.

[175] Oliver E. Williamson. *Markets and Hierarchies*. New York: Free Press, 1975.

[176] Christian Wirth, Johannes Fürnkranz, and Gerhard Neumann. "Model-free preference-based reinforcement learning". In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. 2016, pp. 2222–2228.

[177] Mark Woodward, Chelsea Finn, and Karol Hausman. "Learning to interactively learn and assist". In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. 2020, pp. 2535–2543.

[178] Sarah A. Wu et al. "Too many cooks: Coordinating multi-agent collaboration through inverse planning." In: *Proceedings of the Cognitive Science Society*. 2020.

[179] Daniel Zhang et al. *The AI Index 2021 Annual Report*. URL: `arxiv:2103.06312`.

[180] Simon Zhuang and Dylan Hadfield-Menell. "Consequences of Misaligned AI". In: *Advances in Neural Information Processing Systems* 33 (2020).

[181] Brian D. Ziebart et al. "Maximum Entropy Inverse Reinforcement Learning". In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*. 2008, pp. 1433–1438.