Synthesis of Interpretable and Obfuscatory Behaviors in Human-Aware AI Systems

by

Anagha Kulkarni

A Dissertation Presented in Partial Fulfillment
of the Requirement for the Degree
Doctor of Philosophy

Approved March 2021 by the
Graduate Supervisory Committee:

Subbarao Kambhamapti, Chair
Ece Kamar
David E. Smith
Siddharth Srivastava
Yu Zhang

ARIZONA STATE UNIVERSITY

May 2021

ABSTRACT

In settings where a human and an embodied AI (artificially intelligent) agent coexist, the AI agent has to be capable of reasoning with the human's preconceived notions about the environment as well as with the human's perception limitations. In addition, it should be capable of communicating intentions and objectives effectively to the human-in-the-loop. While acting in the presence of human observers, the AI agent can synthesize interpretable behaviors like explicable, legible, and assistive behaviors by accounting for the human's mental model (inclusive of her sensor model) in its reasoning process. This thesis will study different behavior synthesis algorithms which focus on improving the interpretability of the agent's behavior in the presence of a human observer. Further, this thesis will study how environment redesign strategies can be leveraged to improve the overall interpretability of the agent's behavior. At times, the agent's environment may also consist of purely adversarial entities or mixed entities (i.e. adversarial as well as cooperative entities), that are trying to infer information from the AI agent's behavior. In such settings, it is crucial for the agent to exhibit obfuscatory behavior that prevents sensitive information from falling into the hands of the adversarial entities. This thesis will show that it is possible to synthesize interpretable as well as obfuscatory behaviors using a single underlying algorithmic framework.

To my late father, Pradeep Kulkarni,

whose love continues to be a pillar of stability,

and to my mother, Seema Kulkarni,

whose kindness provides me constant warmth

ACKNOWLEDGEMENTS

A number of people have helped me make this dissertation possible. First and foremost, I would like to thank my advisor Prof. Subbarao Kambhampati. This work would not have been possible without his support and guidance. He has supported me through my highs and especially through my lows. I admire his ability to draw deep insights from problems and to simplify them by hacking away at fluff. I also admire the passion and energy with which Prof. Kambhampati conducts himself. He is a kind of an advisor who will always make time for his students despite his busy schedule – be it weekends or late evenings. I consider myself lucky to have had somebody as energetic and dynamic as him as my advisor. I am also very thankful to him for the various opportunities he has granted me, such as encouraging me to present a conference tutorial alongside him and encouraging me to contribute to a book on my research work. I will be forever grateful to him for the time and energy he has invested in me.

I would also like to thank Prof. Siddharth Srivastava for his constant guidance and patience. I have spent a numerous hours refining my half-baked ideas and scribbling them on his wall (which doubled as a whiteboard). I admire the regularity and discipline with which he conducted our one-on-one meetings. I thank him for unknowingly instilling those values in me. I interacted most with Prof. Yu Zhang during the first couple years of my PhD. It was a great help to have him as a sounding board to discuss ideas. While I collaborated with him, I got to work on quite a few fun robot demonstration projects, which made the initial part of my Ph.D. very exciting. I am thankful to him for his constant guidance during those years. I am happy to have gotten a chance to collaborate with Dr. David E. Smith. I admire his ability to look at a problem and bring out a cogent, generalized perspective to it. All of his visits to ASU have lead to interesting discussions, which have evolved into interesting

works. I am also thankful to Dr. Smith for sending us some amazing holiday snacks including his signature dried persimmons over the past few Decembers. Finally, it has been helpful to have a role model like Dr. Ece Kamar to learn and grow from. I first met her at a women's mentoring program at AAAI 2020. Even in this day and age, women are still a minority in many STEM Ph.D. programs and it is helpful to have people like her who are willing to give back to the community by mentoring young people.

Moreover, I have been lucky to have gotten a chance to work with many other collaborators: Dr. Satya Gautam Vadlamudi who helped me get my bearings in the lab when I was a new Ph.D. student, Dr. Sarah Keren whose guidance has been invaluable, and who evolved from being a good collaborator to a good friend, Dr. Matthew Klenk and Dr. Shiwali Mohan from Palo Alto Research Center (PARC) who made me feel welcome at PARC during my 2018 summer internship. I would also like to thank Prof. Satish Kumar Thittamaranahalli (T. K. Satish Kumar) from University of Southern California for contributing to my journey. Without his recommendation of getting in touch with Prof. Kambhampati for the Ph.D. program, I wouldn't have started this journey in the first place.

I would also like to thank all of my wonderful labmates, who have made this journey a sweet and a memorable one. Sarath and Tathagata, specifically, have been two of the best collaborators throughout my time at the lab – right from my first project on explicable planning to the Imagine Cup competition in Seattle to the tutorial presentation at a conference in my final year. Lydia, Sailik and Sachin have been my constant companions throughout this journey, helping me become a better version of myself. And Yantian who has helped me become a fitter version of myself. Alberto, Zahra and Niharika who sat beside me in the lab, and made me look forward to coming and working in the lab. Sriram, Utkarsh, Lin, Mudit, Karthik and Siddhant

who have all made the lab a fun, energetic and enjoyable place.

Last but not the least, I would like to thank my family members for supporting me throughout these past six years. I will be forever grateful to my partner and soulmate, Rohan. Without his constant support and encouragement, I could not have made so many of the paper deadlines. He has been there for me every step of this journey. In fact, without his encouragement, I probably would not have embarked on this journey in the first place. I am also grateful to my parents who have always put my problems ahead of theirs and supported my decisions no matter what. It breaks my heart to realize that my father is not here to see me graduate this year. Last year throughout the COVID-19 lockdown, I used to have long conversations with him about my impending graduation and future plans. I wish he was here to see me now, I know he would have been proud of me. I am also thankful to my dear mother, who has always strived to make sure I got the very best of everything. Without a second thought to herself, she has always comforted me with her endless love and support. And finally my younger brother, whom I have always looked up to and who inspires me to be a better version of myself.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

The recent advances in the field of AI have made AI systems a part and parcel of our day-to-day lives. So much so that, each and every routine activity in our daily lives revolves around AI-backed platforms – from a smart speaker powered by an AI assistant in your living room to a smart cooker in your kitchen to a smart toothbrush in your bathroom to a smart car in your garage. The advances in AI have changed the landscape of various industries like finance, healthcare, marketing, education, etc. However most of these AI platforms are not *human-aware*. They are not human-aware because they do not account for the human's preconceived notions or intentions while performing an activity. For example, a smart car that expects the human driver to take control of the steering wheel during an event without accounting for the human's response time to that event is not human-aware. A future where humans and AI systems cohabitate and collaborate with true synergy is yet to come!

For an AI system to be truly human-aware, there are quite a few considerations that need to be taken into account. First of all, the AI system should have a sense of the human's understanding of the task. Further, it should use this information to synthesize behavior that the human expects the system to generate. A human-aware AI system may have to take the role of a teacher or of a teammate or of a subordinate depending on the task at hand and depending on the human's capabilities and limitations. Some of the application domains that may directly benefit from a true human-AI synergy involve commercial settings (like factory floors [18; 17], warehouses, restaurants [58], etc.) where humans and robots are working alongside each other as co-workers in shared workspaces (as against robots operating in a separate workspace

to avoid accidents), or in disaster response settings [4; 16] (like collapsed or unstable structures) where it is physically unsafe for the humans to operate but the robot can collaborate remotely with the humans to perform certain tasks, or in decision support settings (like providing assistance/advise to pilots in the cockpit [5], providing suggestions to doctors in a clinical setup [2; 3]) where the AI agent is helping the human counterpart perform a computationally challenging task, etc. As it can be seen, there are many industries that may benefit from true human-aware AI agents.

However, not all applications of human-aware AI fall under settings with cooperative interactions. There are also some application domains that require the AI agent to tackle adversarial entities. In such situations, it is equally important for the AI agent to model the adversarial entity and to respond in a way that minimizes the leakage of sensitive information. Therefore, the AI agent has to play the role of a privacy preserving agent. For instance, there are directly relevant scenarios like military planning where the enemies may be interested in tracking say, the troop movements to discover some sensitive information, or more commonly day-to-day scenarios, involving the supermarket or department stores that use blue-tooth beacons placed in aisles to gather information about the customer by pinging her mobile phone bluetooth without her knowledge. In such situations, the AI agent should have a sense of the adversary's objective of gleaning sensitive information and should ensure that the user's information leakage is at a minimum.

This thesis will establish a taxonomy of different types of human-aware AI behaviors (as shown in Figure 1.1) and will explore and discuss how these behaviors can be synthesized and the challenges encountered in synthesizing them. In this thesis, I will consider problem settings where an AI system is an autonomous embodied robotic agent with the objective of solving a task in the presence of some human observers. These human observers due to their vested interest in the robot's objectives, may

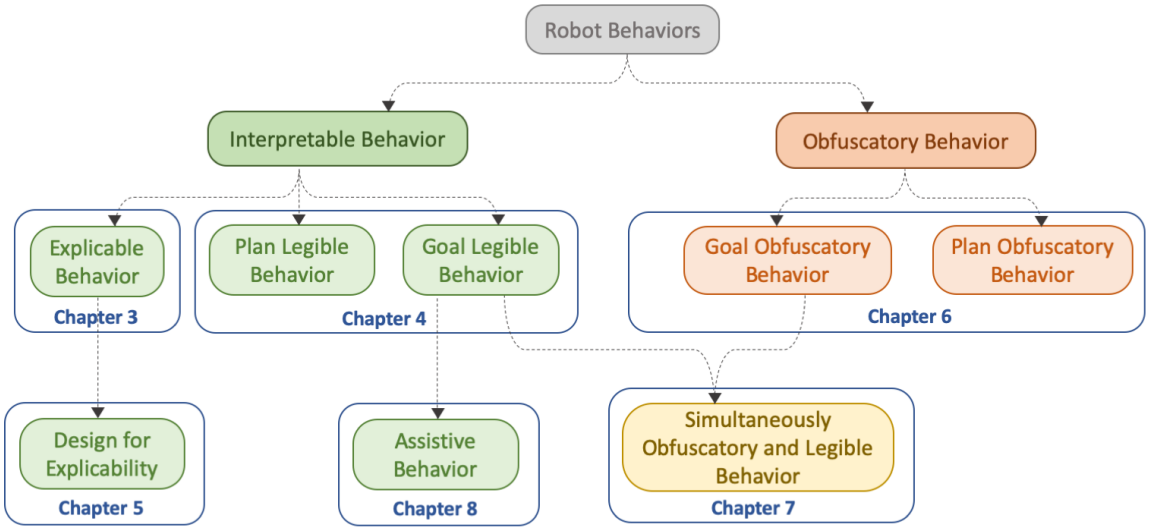


Figure 1.1: An Overview of the Thesis.

be inclined to try and infer the behavior of the robot. The humans in the environment may have limited perception of the robot's activities or may have incomplete or inaccurate beliefs about the robot's capabilities, beliefs or intentions. The robot has to be capable of reasoning over these aspects and synthesize behaviors that are interpretable to the humans as well as assistive despite their limited perception or computational capabilities. On the flip side, if the observing entity has adversarial intent towards the robot, the robot has to synthesize behaviors that restrict the information inferred by the adversarial observer. In addition, if there are multiple human observers in the environment with varying relationship (cooperative as well as adversarial) with the robot, it has to synthesize behaviors that balance the amount of information inferred by each of the observers. Furthermore, this thesis will also explore the benefits of designing an environment in which the robot is operating in order to engender desirable behavior from the robot.

1.1 Thesis Overview

With the above discussion in mind, the thesis (refer Figure 1.1) is organized as follows:

- **Chapter 2** lays out a brief background on the type of problems that will be discussed in this thesis. Specifically, this chapter will discuss the preliminary notations that will be used to represent the robot's model as well as the human's mental model and her perception limitations, which are crucial in the synthesis of human-aware robot behaviors.

- **Chapter 3** will discuss the synthesis of a type of interpretable robot behavior namely explicable behavior. Through explicable behavior the robot can ensure that its behavior aligns with the human's expectations of it. Specifically, we will see two different approaches (model-based and model-free) that can be used to generate such explicable plans.

- **Chapter 4** will focus on another type of interpretable robot behavior namely legible behavior. Through legible behavior the robot can convey some information about its goals or plans to the human observer. Here we will see the controlled observability planning framework that allows the agent to exhibit two types of legible behaviors.

- **Chapter 5** will build on the discussion about explicable behavior. Specifically, it will focus on leveraging environment design techniques to optimize the environment in a way that boosts explicable robot behaviors. We will also see a way to model explicability given a longitudinal interaction between the human and the robot. Further, we will briefly discuss the notion of environment design for legible and predictable behaviors and how it connects to the existing literature on goal/plan recognition design.

- **Chapter 6** will discuss the synthesis of obfuscatory behaviors which allow the robot to hide information about its goals and plans. Here we will see how the

goal obfuscation approach can maintain obfuscation even when the algorithm is queried with different inputs (i.e. how it can be secure to replay attacks). We will also see how the robot can modulate the coverage of goal obfuscation depending on its available resource budget.

- **Chapter 7** will discuss a more general scenario where the environment can consist of both adversarial as well as cooperative observers. Here we will see how the robot can optimally balance the amount of goal obfuscation for an adversarial observer with the amount of goal legibility for a cooperative observer.

- **Chapter 8** will discuss the synthesis of proactively assistive behaviors. With proactive assistance, it is important for the robot to ensure that the human observer is aware of the potential cost reduction. Here we will see an extension of the controlled observability planning framework which models a human observer who not only can observe but also act in the environment.

Chapter 9 concludes the thesis with a summary of the various approaches used to synthesize interpretable and obfuscatory behaviors. Here we will reflect on various aspects of the presented work as well as avenues for future directions and highlight the key takeaways of this thesis.

Chapter 2

BACKGROUND

Before we dive into each of the aforementioned robot behaviors, we will first take a look at some of preliminary notations and concepts that will help us in setting the stage for the main thesis contributions.

2.1 Preliminaries

We will be using the notation of planning problems [35] to define the frameworks in this thesis.

2.1.1 Planning

A planning problem can be defined as a tuple $\mathcal{M} = \langle \mathcal{F}, \mathcal{A}, \mathcal{I}, G, c \rangle$, where $\mathcal{F}$, is a set of fluents, $\mathcal{A}$, is a set of actions, and $c : \mathcal{A} \to \mathbb{R}_{>0}$ is the cost for executing an action. A state s of the world is an instantiation of all fluents in $\mathcal{F}$. Let $\mathcal{S}$ be the set of states. $\mathcal{I} \in \mathcal{S}$ is the initial state, that is all the fluents are instantiated. G is a goal condition where a subset of fluents in $\mathcal{F}$ are instantiated. Each action $a \in \mathcal{A}$ is a tuple of the form $\langle pre(a), add(a), del(a) \rangle$ where $pre(a) \subseteq \mathcal{F}$ is a set of preconditions, $add(a) \subseteq \mathcal{F}$ is a set of add effects and $del(a) \subseteq \mathcal{F}$ is a set of delete effects of action a. The transition function $\Gamma_{\mathcal{M}}(\cdot)$ is given by $\Gamma_{\mathcal{M}}(s, a) \models \perp$ if $s \not\models pre(a)$; else $\Gamma_{\mathcal{M}}(s, a) \models s \cup add(a) \setminus del(a)$. The solution to $\mathcal{M}$ is a *plan* or a sequence of actions $\pi = \langle a_1, a_2, \ldots, a_n \rangle$, such that, $\Gamma_{\mathcal{M}}(\mathcal{I}, \pi) \models G$, i.e., starting from the initial state and sequentially executing the actions results in the robot achieving the goal. The cost of the plan, $c(\pi)$, is a sum of the cost of all the actions in it, $c(\pi) = \sum_{a_i \in \pi} c(a_i)$.

2.1.2 Human-aware Planning

An autonomous robot's planning model $\mathcal{M}^R = \langle \mathcal{F}, \mathcal{A}^R, \mathcal{I}^R, G^R, c^R \rangle$, where $\mathcal{F}$ is a set of fluents, $\mathcal{A}^R$ is a set of its actions, $\mathcal{I}^R$ is its initial state, G^R is its goal and c^R is its cost function, can compute a sequence of actions from its initial state that optimizes its cost (resource or time) towards its goal. But when the robot is operating in the presence of a human observer who has vested interests in the robot's goal, the robot cannot simply reason with its own model or optimize its own cost to the goal. In order to ensure it is being human-aware, the robot should model the human's beliefs and understanding of the robot model captured as $\mathcal{M}_h^R = \langle \mathcal{F}, \mathcal{A}_h^R, \mathcal{I}_h^R, G_h^R, c_h^R \rangle$, representing the human's mental model of the robot model. Note that this is different from $\mathcal{M}^H = \langle \mathcal{F}, \mathcal{A}^H, \mathcal{I}^H, G^H, c^H \rangle$, which represents the human's model of her own task.

Further, the human observer who is observing the robot may sometimes have imperfect observations of the robot's activities. In this thesis, we will study some of the complications resulting from human's perception limitations captured using the parameters of her sensor model. The human's sensor model can be represented as, $Obs^H = \langle \Omega^H, \mathcal{O}^H \rangle$, where Ω^H is a set of observation tokens that are distinguishable by the human. We will use $\omega \in \Omega^H$ to denote an observation token, and $\langle \omega_1, \ldots, \omega_n \rangle$ to denote an observation sequence. $\mathcal{O}^H$ is an observation function that maps an action performed and next state achieved to an observation token in Ω. This function captures any preception limitations present in the human's sensor model. Given a sequence of tokens, $\langle \omega_1, \ldots, \omega_n \rangle$, a plan π is consistent with this sequence if and only if $\langle \omega_1, \ldots, \omega_n \rangle \models \pi$. Whenever the human's sensor model produces imperfect observations, it can be taken into account as part of her mental model, written as, $\mathcal{M}_h^R = \langle \mathcal{F}, \mathcal{A}_h^R, \mathcal{I}_h^R, G_h^R, c_h^R, Obs^H \rangle$.

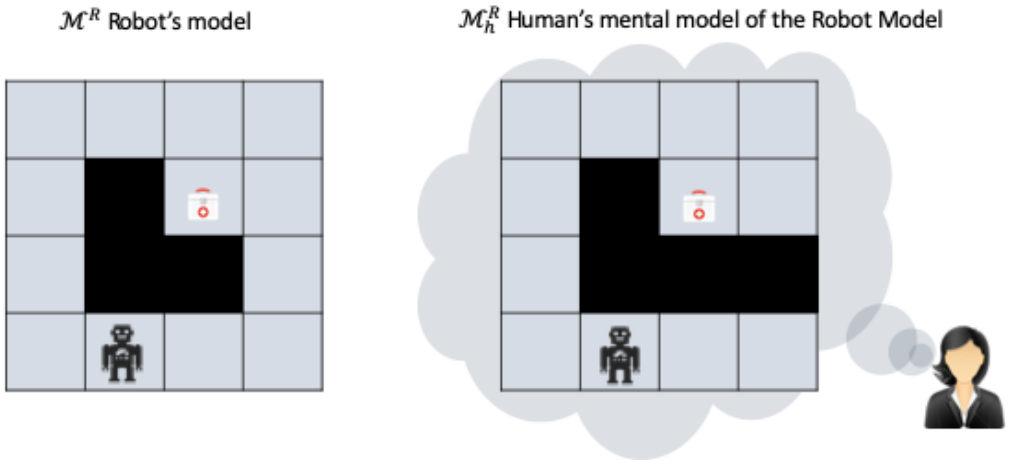


Figure 2.1: Illustration of Differences Between the Robot's Model and the Human's Mental Model of the Robot in a Urban Search and Rescue Domain.

We will now see some reasons for a drift between the robot's model and the human's understanding of it, and how different aspects regarding the human's biases and the human's sensor model of the robot's activities affect the human's mental model of the robot model.

2.1.3 Human's Mental Model of the Robot Model

The human observer may have partial or inaccurate understanding of how the robot operates in the environment and/or what goals/plans the robot is intending to follow in the environment. The robot should be able to model the human's misconceptions and beliefs about itself in the form of a mental model of the human. The human's mental model maybe different from the robot's actual model in multiple ways. It maybe different in terms of robot's capabilities. The human might have an incorrect understanding of the robot's actions or may incorrectly attribute capabilities to the robot that are not physically possible for the robot. Or she may have incorrect understanding of the state of the environment. She may assume certain

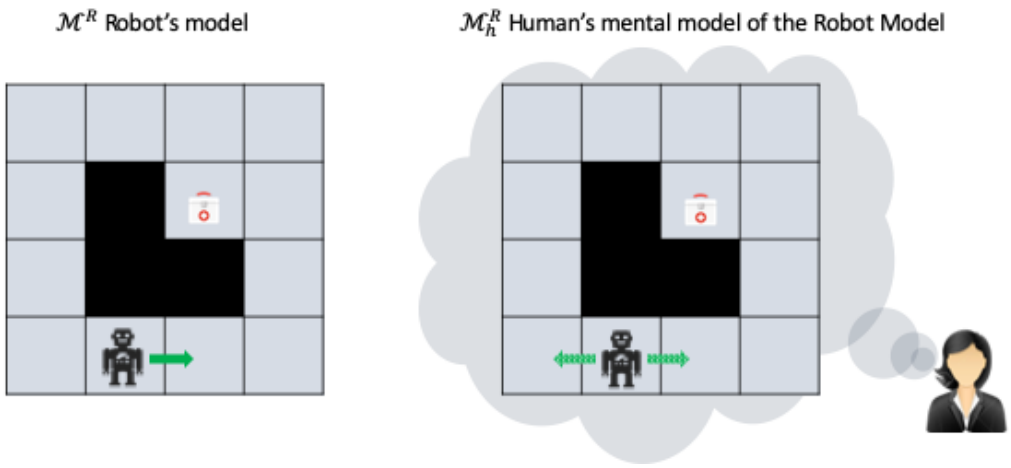


Figure 2.2: Illustration of the Impact of Human's Noisy Sensor Model on Human's Mental Model of the Robot in a Urban Search and Rescue Domain.

facts in the environment are true when they are not in actuality. She may also have incorrect assumptions about the robot's objectives or intentions. She may have partial observability of robot's actions or may have incorrect information about robot's sensor capabilities. She may also be reasoning in terms of a different or a simpler representation than the one robot is using. Due one or many of these reasons the robot cannot simply optimize its behavior with respect to its own model, and instead may have to also account for the human's mental model while synthesizing its behavior. Figure 2.1 illustrates the differences between the robot's model and the human's understanding of the robot model. This example shows a urban search and rescue scenario where the internal agent i.e. the robot has a different map than the external agent i.e. the human commander. Therefore, the robot's optimal path to the medkit is uninterpretable to the human.

Perception Limitations of the Human

Even when the human has complete and correct understanding of the robot's capabilities, she may suffer from partial observability of the robot's activities. For instance, the human may have access to a coarse-grained GPS sensor model that cannot distinguish between robot's movement within a 10 ft radius. That is the human's sensor model, Obs^H, may not be fine-grained (i.e. a noisy sensor model) such that a single observation about the robot's activity may map to multiple similar looking activities plausibly performed by the robot. Or there maybe some activities that do not get recognized by the human's sensor model (i.e. some activities maybe non-observable). Or there maybe certain time steps at which activities do not get recorded (i.e. some time steps may have missing observations). These kinds of partial observations require the human to maintain a belief about the robot's activities. That is the partial observability results in divergence of human's belief from the actual robot state. As a result, the robot cannot simply optimize its behavior with respect to its own model, and may instead have to also account for the human's sensor model while synthesizing its behavior. Figure 2.2 illustrates the uncertainty about robot's motion in human's mental model due to human's noisy sensor model. Here the robot and the human have synced up on the current map of the environment, although due to noisy observations, it is not clear to the human whether the robot has moved left or right.

2.2 Implicit Communication through Behavior

In the plan recognition literature, depending on the role played by the robot, the process of inference of the robot's activities has been classified into two major categories, namely, keyhole recognition and intended recognition [23; 11]. In keyhole recognition, the robot performs its activities without the intention of impacting the

inference process of the observers. In contrast, in intended recognition, the robot is aware of the observer's model and performs activities to either actively aid or hinder the process of inference. In this thesis, we consider a robot of the latter type.

A robot that is aware of the human observer's model can choose to communicate the information it wants the humans to know. However, explicit communication comes with added communication cost and constraints. Explicit communication requires multiple considerations: what information should be communicated, when should it be communicated, how should it be communicated, etc. In particular, explicit communication may suffer from vocabulary mismatch between the robot and the human. For instance, the robot may be operating at a more fine-grained level of information, while the human may be operating at an abstract level. Explicit communication may also suffer from inherent delays in communicating information. For instance, consider a Mars rover, it takes about 22 minutes for the communication signals to reach from Mars to Earth and back. Moreover, it might not be always possible for the robot to communicate explicitly. For instance, in a disaster response scenario, there might not be an explicit facility to enable communication, instead the human observer may only be able to see the movement of the robot on a map. In such cases, it becomes important for the robot to actively alter the inference process of the human by synthesizing behavior that implicitly communicates necessary information to the human.

In this thesis, we broadly explore two types of behaviors that achieve implicit communication: interpretable behaviors that aid a cooperative observer's inference process while obfuscatory behaviors that hinder an adversarial observer's inference process. Figure 2.3 illustrates the difference between interpretable and obfuscatory behaviors.

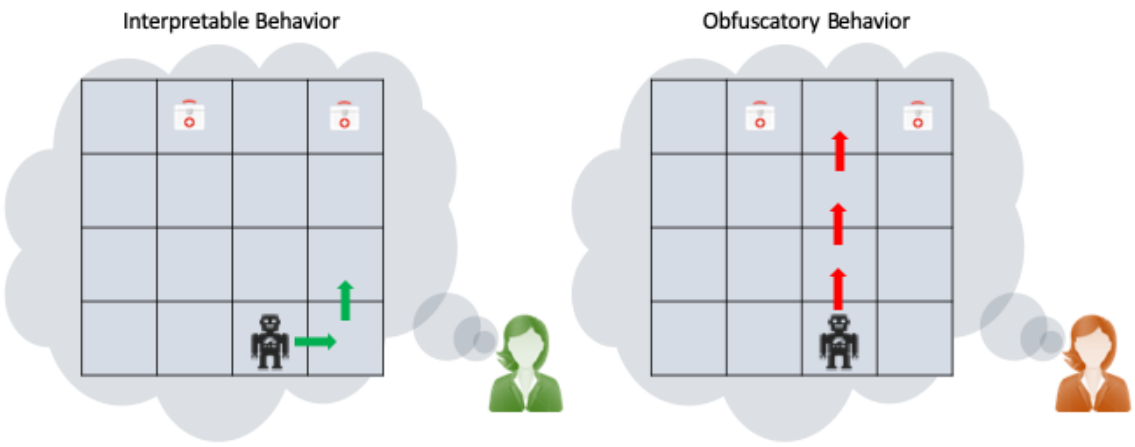


Figure 2.3: Illustration of Implicit Communication for Cooperative as Well as Adversarial Observers. Through Behavior, the Robot Can Communicate Its Goal of Picking up the Rightmost Medkit to the Cooperative Observer in the Leftmost Sub-figure, While It Can Hide Its Intentions from the Adversarial Observer in the Rightmost Sub-figure by Taking an Ambiguous Path.

2.2.1 Interpretable Behavior

When the robot is acting in the presence of a cooperative human observer, it has to ensure that its decisions are interpretable to the human-in-the-loop. Uninterpretable behavior can lead to increased cognitive load on the human – from reduced trust and productivity to increased risk of danger around the robot [32]. The *Roadmap for U.S. Robotics* [21] emphasizes – *"humans must be able to read and recognize robot activities in order to interpret the robot's understanding"*. The robot's behavior may be uninterpretable if the human: (1) has an incorrect notion of the robot's beliefs and capabilities [102; 19; 57] (2) is unaware of the robot's goals and rewards [27; 59] or (3) cannot predict the robot's plan or policy [34; 59].

Thus, in order to be interpretable, the robot must take into account the human's expectations of its behavior – i.e. *the human mental model*. In this thesis, we will

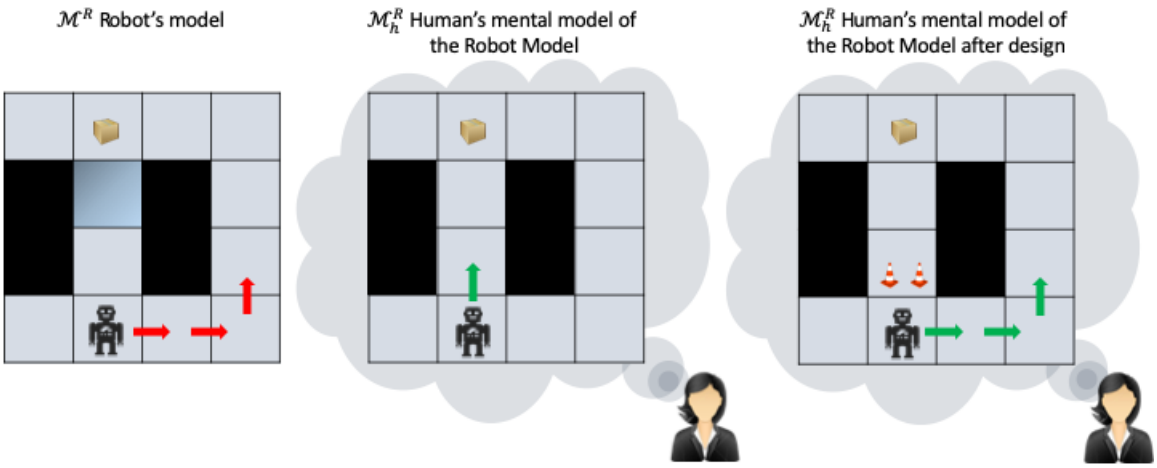


Figure 2.4: Illustration to Demonstrate Improvement in Explicability of the Robot's Behavior after Environment Design.

discuss the synthesis of the three main types of interpretable behaviors: (1) explicable behavior allows the robot to conform to human's expectations, (2) legible behavior allows the robot to communicate information about its goals and plans to the human and (3) interpretable assistive behavior that allows the robot to communicate how the assistance affects the human's future decisions. These behaviors will be covered in detail in Chapters 3, 4 and 8 respectively.

Further, we will also see how the interpretability associated with achieving certain tasks can be improved by leveraging environment design. One advantage of redesigning the environment is that the process of generation of interpretable behavior is offloaded from the robot onto the design process. In other words, the computational overhead of generating interpretable behavior – dealing with the human mental model and reasoning over it – becomes a part of the design process, which can be done offline. Figure 2.4 illustrates this concept. Consider a robot which cannot pass through a passageway because of a reflective surface. This might not be evident to the human teammates who find the roundabout route of the robot (in the leftmost subfigure)

inexplicable. After installing barricades (as shown in the rightmost subfigure), it becomes evident to the humans that the robot cannot pass through the passageway. Thus alleviating their worries and increasing the explicability of the robot's behavior. This notion will be covered in detail in Chapter 5.

2.2.2 Obfuscatory Behavior

When the robot is acting in the presence of an adversarial observer, it has to ensure that its decisions do not reveal sensitive information about its task. If the robot's activities are not secure, an adversarial observer can use diagnosis to infer private information and interfere with the robot's objectives. For instance, in military planning, adversaries observe troop movements to infer possible targets; in corporate strategy, competitors predict each others future directions by observing potential partnerships; in product design, component specifications often portend new product's functionality.

Thus, in order to preserve the privacy of sensitive information, the robot has to take into account the adversary's mental model. In this thesis, we will discuss the synthesis of three types of obfuscatory behaviors: (1) goal obfuscatory behavior that allows the robot to hide sensitive information about its objectives, (2) plan obfuscatory behavior that allows the robot to hide information about its activities from the adversarial observer, and (3) secure goal obfuscatory behavior that allows the robot to obfuscate information even when the adversary has access to the robot's behavior generation algorithm. Further, we will also see behavior synthesis challenges when both the adversarial as well as the cooperative observers exist in the environment. In such a setting, with multiple different observers, the objective of the robot is to provide necessary information to the cooperative observers while hiding it from adversarial observer. We will explore these behaviors in detail in Chapter 6 and 7.

Chapter 3

PLANNING FOR EXPLICABLE BEHAVIOR

In this chapter, we will focus the discussion on explicable behavior which is a type of interpretable behavior. An important challenge in the human-in-the-loop scenarios is to ensure that a robot's behavior is comprehensible to the humans in the loop. Without it, the robot runs the risk of increasing the cognitive load of humans which can result in reduced productivity, safety, and trust [32]. The robot's behavior may seem inexplicable to a human when there is a mismatch between the robot's plans and the human's expectations of the robot's plans. This mismatch may arise because of the difference in the actual robot model $\mathcal{M}^R$, and the human's mental model of it, $\mathcal{M}^R_h$. For example, consider a scenario with an autonomous car switching lanes on a highway. The autonomous car, in order to switch the lane, may make sharp and calculated moves, as opposed to gradually moving towards the other lane. These moves may well be optimal for the car due to its superior sensing and steering capabilities. Nevertheless, a passenger sitting inside may perceive this as dangerous and reckless behavior.

In order to avoid being inexplicable, the robot has to reason with the human's mental model and compute the plans that align with this model. As long as the robot's behavior is aligned with the human mental model, the human can make sense of it. Therefore, the objective of explicable planning is to generate robot plans that not only minimize the cost of the plan, but also the distance between the robot plan produced by $\mathcal{M}^R$ and the plan expected by the human produced by $\mathcal{M}^R_h$. Of course, an immediate question is, if $\mathcal{M}^R_h$ is available to the robot, why is $\mathcal{M}^R$ required in the plan generation process at all? We note that this is a necessary component

since the human mental model might entail plans that are not even feasible for the robot or are prohibitively expensive, and can thus at best serve as a guide, and not an oracle, to the explicable plan generation process. Therefore, instead of using $\mathcal{M}_h^R$ directly, the robot can use $\mathcal{M}_h^R$ as a guide to compute plans that reduce the distance with human's expected plans. In settings where the objective is to minimize the cognitive load on the human or minimize the cost of explicit communication of explanations [19; 63], the computation of explicable plans can be crucial. Also, settings where the observers are not necessarily experts in the domain and tend to have noisy or incomplete understanding of robot behavior, explicable plans can be useful for engendering trust.

An important consideration in the computation of explicable behavior is access to the human mental model, $\mathcal{M}_h^R$. We present two settings in this chapter, one where the robot has access to $\mathcal{M}_h^R$, i.e. model based explicable planning, and another where the robot learns an approximation of it, $\widehat{\mathcal{M}_h^R}$, i.e., model-free explicable planning. In many domains, such as in factory scenarios, mission planning or household, there is generally a clear expectation of how a task should be performed. In such cases, $\mathcal{M}_h^R$ can be constructed following the norms or protocols that are relevant to that domain. Most deployed products make use of inbuilt models of user expectations in some form or the other. Building such models, of course, require interactions with users of that domain.

In model-based explicable planning, we hypothesize that the plan distances [93; 74] can quantify the distance between the robot plan $\pi_{\mathcal{M}^R}$ and the expected plans $\pi_{\mathcal{M}_h^R}$ from $\mathcal{M}_h^R$. The domain modeler constructs $\mathcal{M}_h^R$, which is then used to generate expected plans. Then we compute the distance between plans from $\mathcal{M}^R$ and $\mathcal{M}_h^R$. The test subjects provide explicability assessments (scores reflecting explicability) for robot plans. Then the scores are mapped to the precomputed distances and a

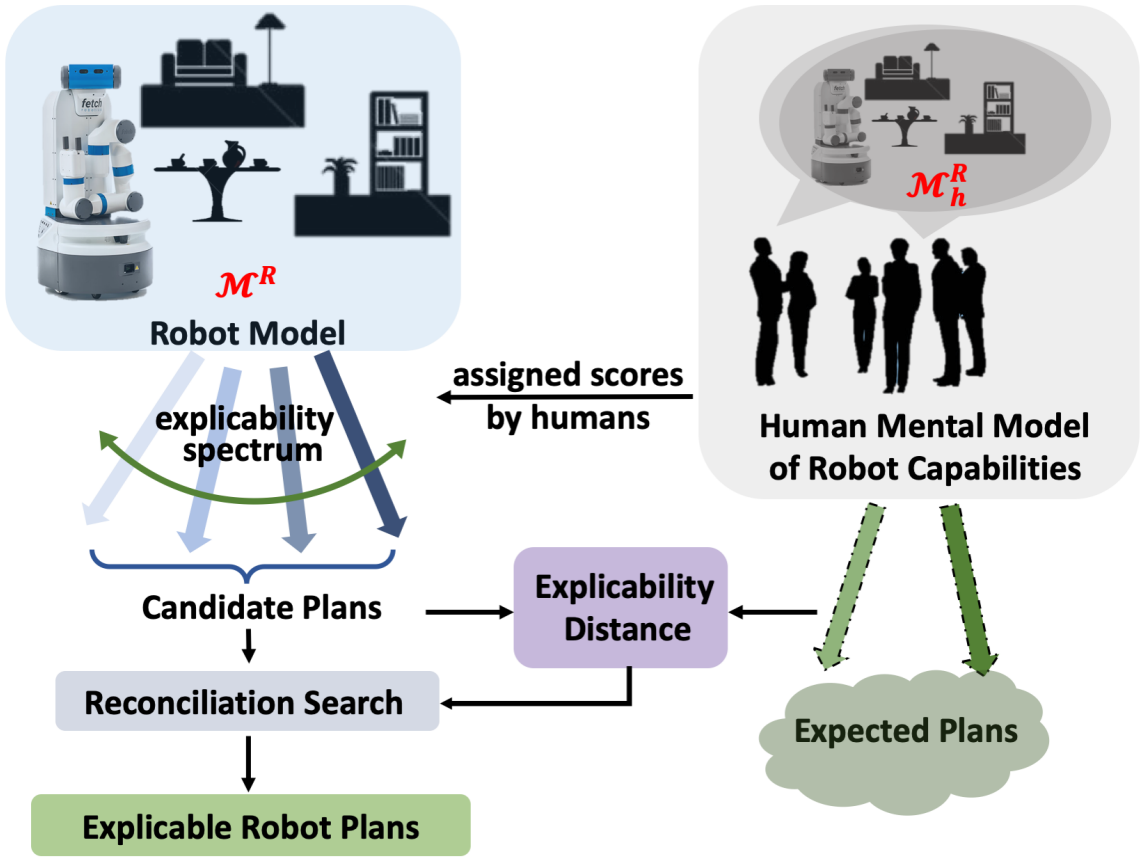


Figure 3.1: Schematic Diagram of the Setting: Here a Regression Model Called Explicability Distance Is Learned to Fit Plan Scores Assigned by Humans to Plan Distances Between the Robot's and the Human's Expected Plans. This Gives a Heuristic for Computing Explicable Plans, Which Is Used by the Reconciliation Search.

regression model of the explicability distance is learned. The plan generation process uses this learned explicability distance as a heuristic to guide the search. This process is illustrated in Figure 3.1.

In model-free explicable planning, we learn an approximation of the human mental model represented as $\widehat{\mathcal{M}_h^R}$. This model is learned based on an underlying assumption that humans tend to associate tasks/sub-goals with actions in a given plan [96; 25]. The labeling scheme used by test subjects, to associate domain-specific task labels to

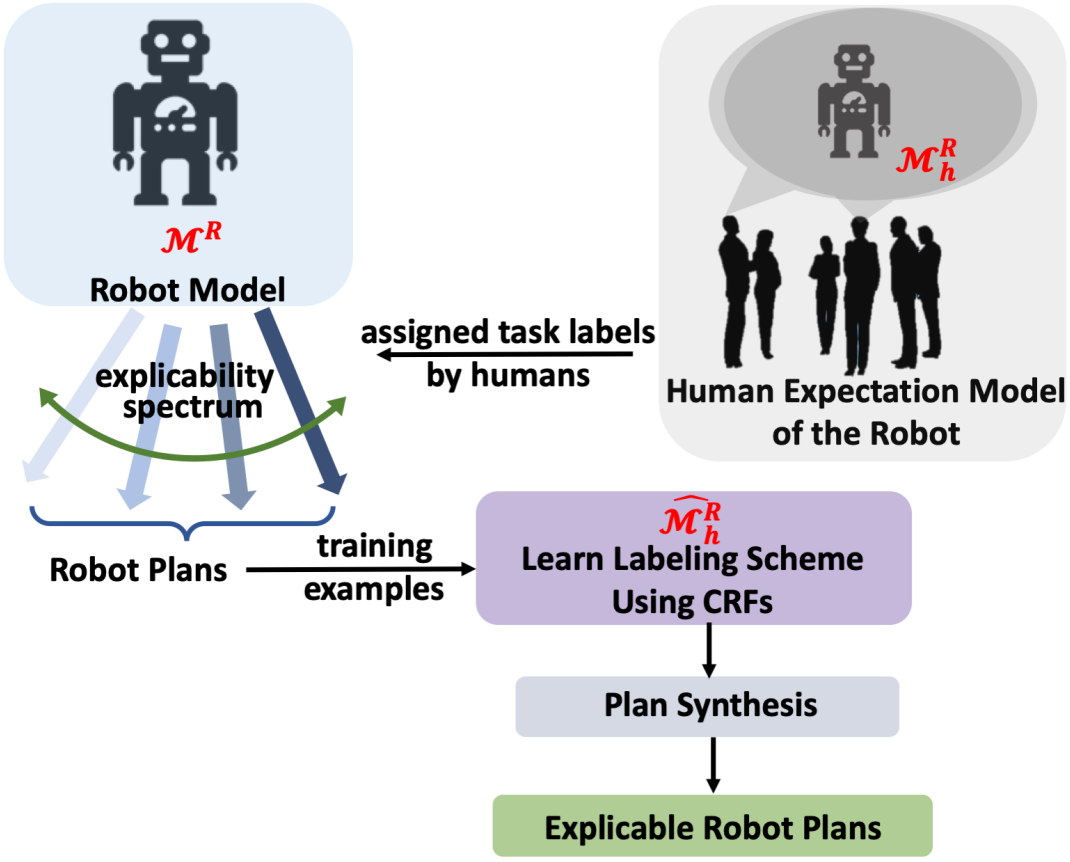


Figure 3.2: Schematic Diagram of the Second Setting. Here the Conditional Random Field (CRFs) Are Used to Learn a Labeling Scheme over the Task Labels Assigned by Humans to Robot Plans. This Model Gives the Heuristic for Computing Explicable Plans.

actions in the plan, is used to learn an approximation $\widehat{\mathcal{M}_h^R}$. This labeling scheme is learned from training examples provided by test subjects using conditional random fields (CRFs). This learned model is then used as a heuristic to generate explicable

plans. This approach is illustrated in Figure 3.2.

3.1 Related Work

An important requirement for a robot collaborating with a human is the ability to infer human's goals and intents and use that information to guide its planning process. There have been various efforts in the direction of human aware planning to account for human's beliefs and goals [87; 22; 67; 12; 20] to encourage human robot interactions. More recently, the efforts have been directed towards studying and understanding various forms of interpretable robot behaviors, especially in the task planning and motion planning communities. Specifically, a recent survey [14] on various interpretable behaviors establishes a taxonomy over the different types of interpretable behaviors namely: explicability, legibility and predictability and provides coherent categorization of recent works in this direction. This chapter specifically focuses on the generation of explicable behavior, while Chapter 4 deals with the notion of legibility. Legible behaviors emerged as a way to reduce ambiguity over the goals of the robot [28; 54]. However, these earlier works on legibility did not explicitly consider differences between the robot model and the human mental model of the robot model. Our work on explicable planning explicitly models such differences that can result from the human's assumptions and preconceived notions about the task as well as the robot's capabilities. Further, explicable planning specifically focuses on aligning the robot's behavior with the human's expectations of it, despite the model differences.

The work on explicable planning is also closely connected to the work on generating explanations through "model reconciliation" [19; 89; 92; 91]. The generation of explicable plans and explanations can be seen as complementary strategies to deal with model differences. In the case of explanations, the robot executes its inexpli-

cable behaviors and then provides an explanation to make its behavior explicable to the human. Here the aim is to explicitly update the human's mental model to align it with the robot's model. However, such explicit communication comes with additional considerations - like what to communicate, when to communicate and how to communicate. As the act of incorporating explanations can increase the human's cognitive load especially when the human is operating in a mission critical scenario or when the communicated explanation involves multiple updates. In such situations, the explicable behaviors are more suitable since they do not lead to increase in the cognitive load as long as the explicable plan is closer to the human's expected plan. Further, in certain domains, where communication constraints can prevent communication of explanations altogether, explicable behavior may be the only alternative available to the robot to build trust and to improve the overall interaction efficacy with the human.

In the following sections, we will formally define the explicability planning problem and present solution methodologies to compute explicable plans in each of the settings. We also present empirical analysis of each setting using physical robot-based domains.

3.2 Explicable Planning Problem

The problem of explicable planning arises when the robot plan, $\pi_{\mathcal{M}^R}$, deviates from the human's expectation of that plan, $\pi_{\mathcal{M}_h^R}$. Here $\pi_{\mathcal{M}^R}$ is the robot's plan solution to the planning problem, $\mathcal{M}^R = \langle \mathcal{F}, \mathcal{A}^R, \mathcal{I}^R, G^R, c^R \rangle$; whereas, $\pi_{\mathcal{M}_h^R}$ is the plan solution considering the human mental model of the robot model, such that, $\mathcal{M}_h^R = \langle \mathcal{F}, \mathcal{A}_h^R, \mathcal{I}_h^R, G_h^R, c_h^R \rangle$. The differences in the human mental model can lead to different plan solutions.

Definition 1. *The* ***explicable planning problem*** *is defined as a tuple* $\mathcal{P}_{Exp} = \langle \mathcal{M}^R, \mathcal{M}_h^R, \delta_{\mathcal{M}_h^R} \rangle$*, where,*

- $\mathcal{M}^R$ *is the robot's planning problem*
- $\mathcal{M}_h^R$ *is the human's mental model of the robot's planning problem*
- $\delta_{\mathcal{M}_h^R}$ *is the distance function that the human uses to compute the distance between her expected plan and the robot's plan*

With respect to $\mathcal{M}^R$ and $\mathcal{M}_h^R$, the action names, preconditions, effects, costs of the actions can be different, along with the initial and goal conditions. The solution to an explicable planning problem is an explicable plan that achieves the goal in the robot's model while minimizing the plan distance from an expected plan in the human's mental model.

Definition 2. *A* ***maximally explicable plan*** *is a plan,* $\pi^*_{\mathcal{M}^R}$*, starting at* $\mathcal{I}^R$ *that achieves the goal* G^R*, such that,* $\underset{\pi_{\mathcal{M}^R}}{\operatorname{argmin}}\, c(\pi_{\mathcal{M}^R}) + \delta_{\mathcal{M}_h^R}(\pi_{\mathcal{M}^R}, \pi_{\mathcal{M}_h^R})$.

In the following sections, we will define two different settings one where the human mental model is known and other where an approximation of it is learned. We will also provide details of how the approach differs in these two settings, in terms of generation of explicable plans.

3.3 Model-based Explicable Planning

In this setting, we quantify the explicability of the robot plans in terms of the *plan distance* between the robot plan $\pi_{\mathcal{M}^R}$ and candidate expected plans $\pi_{\mathcal{M}_h^R}$ from $\mathcal{M}_h^R$. Since the distance function is not directly available to us, we learn an approximation of it using a combination of three plan distances measures. The outline of our approach is illustrated in Figure 3.1. Once both the models $\mathcal{M}^R$ and $\mathcal{M}_h^R$ are obtained, our approach takes the following steps:

1. Firstly, the plan distances between the robot plans and the expected plans are computed using plan the three aforementioned plan distance measures.

2. The human subjects are asked to provide scores for each of the candidate robot plans by labeling each action in the plan with an explicable or inexplicable label.

3. Then the human explicability assessments (scores reflecting explicability) of candidate robot plans are mapped to the plan distance measures in form of regression model called explicability distance.

4. The synthesis of explicable plans is achieved by modifying the `Fast-Downward` [40] planner to incorporate an anytime search with explicability distance as the heuristic. This process results in incrementally more explicable plans.

Background on Plan Distance Measures

We now briefly discuss the three plan distances – action, causal link and state sequence distances – proposed in [93; 74], that we use in this work to capture the explicability distance between plans.

Action distance We denote the set of unique actions in a plan π as $A(\pi) = \{a \mid a \in \pi\}$. Given the action sets $A(\pi_{\mathcal{M}^R})$ and $A(\pi^*_{\mathcal{M}^R_h})$ of two plans $\pi_{\mathcal{M}^R}$ and $\pi^*_{\mathcal{M}^R_h}$ respectively, the action distance is,

$$\delta_A(\pi_{\mathcal{M}^R}, \pi^*_{\mathcal{M}^R_h}) = 1 - \frac{|A(\pi_{\mathcal{M}^R}) \cap A(\pi^*_{\mathcal{M}^R_h})|}{|A(\pi_{\mathcal{M}^R}) \cup A(\pi^*_{\mathcal{M}^R_h})|} \tag{3.1}$$

Here, two plans are similar (and hence their distance measure is smaller) if they contain same actions. Note that it does not consider the ordering of actions.

Causal link distance A causal link represents a tuple of the form $\langle a_i, p_i, a_{i+1} \rangle$, where p_i is a predicate variable that is produced as an effect of action a_i and used as a

precondition for the next action a_{i+1}. The causal link distance measure is represented using the causal link sets $Cl(\pi_{\mathcal{M}^R})$ and $Cl(\pi^*_{\mathcal{M}_h^R})$,

$$\delta_C(\pi_{\mathcal{M}^R}, \pi^*_{\mathcal{M}_h^R}) = 1 - \frac{|Cl(\pi_{\mathcal{M}^R}) \cap Cl(\pi^*_{\mathcal{M}_h^R})|}{|Cl(\pi_{\mathcal{M}^R}) \cup Cl(\pi^*_{\mathcal{M}_h^R})|} \tag{3.2}$$

State sequence distance This distance measure finds the difference between sequences of the states. Given two state sequences $(s_0^R, \ldots, s_n^R)$ and $(s_0^H, \ldots, s_{n'}^H)$ for $\pi_{\mathcal{M}^R}$ and $\pi^*_{\mathcal{M}_h^R}$ respectively, where $n \geq n'$ are the lengths of the plans, the state sequence distance is,

$$\delta_S(\pi_{\mathcal{M}^R}, \pi^*_{\mathcal{M}_h^R}) = \frac{1}{n}\Big[\sum_{k=1}^{n'} d(s_k^R, s_k^H) + n - n'\Big] \tag{3.3}$$

where,

$$d(s_k^R, s_k^H) = 1 - \frac{|s_k^R \cap s_k^H|}{|s_k^R \cup s_k^H|} \tag{3.4}$$

represents the distance between two states (where s_k^R is overloaded to denote the set of predicate variables in state s_k^R). The first term measures the normalized difference between states up to the end of the shortest plan, while the second term, in the absence of a state to compare to, assigns maximum difference possible.

3.3.1 Explicability Distance

We start with a general formulation for capturing a measure of explicability of the robot's plans using plan distances. A set of robot plans are scored by humans such that each action that follows the human's expectation in the context of the plan is scored 1 if explicable, and 0 otherwise. The plan score is then computed as the ratio of the number of explicable actions to the total plan length. A set of expected plans, $\Pi^*_{\mathcal{M}_h^R}$, for the planning problem $\mathcal{M}_h^R$, is a set of optimal cost plans that solve $\mathcal{M}_h^R$, $\Pi^*_{\mathcal{M}_h^R} = \{\pi^{(i)}_{\mathcal{M}_h^R} | i = 1, \ldots, n\}$.

This set of expected plans consists of the plan solutions that the human expects the robot to compute. But these plans are not necessarily feasible in the robot model, $\mathcal{M}^R$. In order to compute the minimum distance between a robot plan and a human's expected plan, we use a following composite distance, which uses all three plan distance measures. It is defined as follows:

Definition 3. *A **composite distance**, δ_{Exp} is a distance between pair of two plans $\langle \pi_{\mathcal{M}^R}, \pi_{\mathcal{M}_h^R} \rangle$, such that,*
$\delta_{Exp}(\pi_{\mathcal{M}^R}, \pi_{\mathcal{M}_h^R}) = ||\delta_A(\pi_{\mathcal{M}^R}, \pi_{\mathcal{M}_h^R}) + \delta_C(\pi_{\mathcal{M}^R}, \pi_{\mathcal{M}_h^R}) + \delta_S(\pi_{\mathcal{M}^R}, \pi_{\mathcal{M}_h^R})||^2$.

But for each robot plan we want to find the minimum distance with respect to the set of human's expected plans. We say a distance minimizing plan in the set of the expected plans is defined as follows:

Definition 4. *A **distance minimizing plan**, $\pi^*_{\mathcal{M}_h^R}$, is a plan in $\Pi^*_{\mathcal{M}_h^R}$, such that for a robot plan, $\pi_{\mathcal{M}^R}$, the composite distance is minimized,*
$\pi^*_{\mathcal{M}_h^R} = \{\pi_{\mathcal{M}_h^R} | \ \underset{\pi_{\mathcal{M}_h^R}}{\operatorname{argmin}} \delta_{Exp}(\pi_{\mathcal{M}^R}, \pi_{\mathcal{M}_h^R})\}$.

Our overall objective is to learn a explicability distance function, i.e. learn a mapping of explicability scores (scoring scheme used by the humans in the loop) to the plan distances between a robot plan and corresponding distance minimizing plan in the set of expected plans. To that end, we define a explicability feature vector as follows:

Definition 5. *An **explicability feature vector**, Δ, is a three-dimensional vector, which is given with respect to a distance minimizing plan pair, $\langle \pi_{\mathcal{M}^R}, \pi^*_{\mathcal{M}_h^R} \rangle$, such that,*
$\Delta = \langle \delta_A(\pi_{\mathcal{M}^R}, \pi^*_{\mathcal{M}_h^R}), \delta_C(\pi_{\mathcal{M}^R}, \pi^*_{\mathcal{M}_h^R}), \delta_S(\pi_{\mathcal{M}^R}, \pi^*_{\mathcal{M}_h^R}) \rangle^T$.

This allows us to learn an explicability distance function, $Exp(\pi_{\mathcal{M}^R} \ / \ \pi^*_{\mathcal{M}_h^R})$, which is essentially a regression function, f, that fits the three plan distances to the total

plan scores, with b as the parameter vector, and Δ as the explicability feature vector, such that,

$$Exp(\pi_{\mathcal{M}^R} \ / \ \pi^*_{\mathcal{M}^R_h}) \approx f(\Delta, b) \tag{3.5}$$

Therefore, a regression model is trained to learn the explicability assessment (total plan scores) of the users by mapping this assessment to the explicability feature vector which consists of plan distances for corresponding plans.

3.3.2 Plan Generation

In this section, we present the details of our plan generation phase. We use the learned explicability distance function as a heuristic to guide our search towards explicable plans.

Reconciliation Search

The solution to an explicable planning problem $\mathcal{P}_{Exp}$ is the set $\mathcal{E}_{Exp}$ of explicable plans (with varying degrees of explicability) in $\mathcal{M}^R$. This is found by performing "reconciliation search" (as detailed in Algorithm 1).

Non-Monotonicity

Since plan score is the fraction of explicable actions in a plan, it exhibits non-monotonicity. As a partial plan grows, a new action may contribute either positively or negatively to the plan score, thus making the explicability distance function non-monotonic. Consider that the goal of an autonomous car is to park itself in a parking spot on its left side. The car takes the left turn, parks and turns on its left indicator. Here the turning on of the left tail light after having parked is an inexplicable action. The first two actions are explicable to the human drivers and contribute positively

Algorithm 1 Reconciliation Search

Input: $\mathcal{M}^R, \mathcal{M}_h^R$, max_cost, and explicability distance $Exp(.\ ,.)$

Output: $\mathcal{E}_{Exp}$

$\mathcal{E}_{Exp} \leftarrow \emptyset$ ▷ Explicable plan set

$open \leftarrow \emptyset$; $closed \leftarrow \emptyset$ ▷ Initialize open and closed lists

$open$.insert$(\mathcal{I}, 0, \text{inf})$

while $open \neq \emptyset$ **do**

  $n \leftarrow open$.remove() ▷ Node with highest $h(\cdot)$

  if $n \models G$ **then**

    $\mathcal{E}_{Exp}$.insert(π s.t. $\Gamma_{\mathcal{M}^R}(\mathcal{I}, \pi) \models n$)

  end if

  $closed$.insert(n)

  for each $v \in$ successors(n) **do**

    if $v \notin closed$ **then**

      if $g(n) + cost(n, v) \leq max\_cost$ **then**

        $open$.insert$(v, g(n) + cost(n, v), h(v))$

      end if

    else

      if $h(n) < h(v)$ **then**

        $closed$.remove(v)

        $open$.insert$(v, g(n) + cost(n,\ v), h(v))$

      end if

    end if

  end for

end while

return $\mathcal{E}_{Exp}$

to the explicability score of the plan but the last action has a negative impact and decreases the score.

Due to non-monotonic nature of explicability distance, we cannot stop the search process after finding the first solution. Consider the following: if e_1 is explicability distance of the first plan, then a node may exist in the open list (set of unexpanded nodes) whose explicability distance is less than e_1, which when expanded may result in a solution plan with explicability distance higher than e_1. A greedy method that expands a node with the highest explicability score of the corresponding partial plan at each step is not guaranteed to find an optimal explicable plan (one of the plans with the highest explicability score) as its first solution. Therefore, to handle the non-monotonic nature, we present a cost-bounded anytime greedy search algorithm called *reconciliation search* that generates all the valid loopless candidate plans up to a given cost bound, and then progressively searches for plans with better explicability scores. The value of the heuristic *h(v)* in a particular state v encountered during search is based entirely on the explicability distance of the robot plan prefix $\pi_{\mathcal{M}^R}$ up to that state,

$$\begin{aligned} h(v) \ &= Exp(\pi_{\mathcal{M}^R}/\pi'_{\mathcal{M}_h^R}) \\ \text{s.t. } &\Gamma_{\mathcal{M}^R}(\mathcal{I}, \pi_{\mathcal{M}^R}) \models v \text{ and} \\ &\Gamma_{\mathcal{M}_h^R}(\mathcal{I}, \pi'_{\mathcal{M}_h^R}) \models v \end{aligned} \tag{3.6}$$

We assume that the same state space is reachable for computation of the plan prefix π' from $\mathcal{I}$ to v in $\mathcal{M}_h^R$ (as per Equation 3.6). We implement this search in the `Fast-Downward` planner. The approach is described in detail in Algorithm 1. At each iteration of the algorithm, the plan prefix of the robot model is compared with the explicable trace $\pi'_{\mathcal{M}_h^R}$ (these are the plans generated using $\mathcal{M}_h^R$ up to the current state in the search process) for the given problem. There are few choices we could

consider in creating such prefixes, such as using optimal or simply valid plans to the state in $\mathcal{M}_h^R$. Using the computed distances, the explicability score for the candidate robot plans is predicted. The search algorithm then makes a locally optimal choice of states. We do not stop the search after generating the first solution, but instead, continue to find all the valid loopless candidate solutions within the given cost bound or until the state space is completely explored.

3.3.3 Evaluation using Simulated Autonomous Car Domain

We evaluated our system using a simulated autonomous car domain. We constructed a human mental model by interviewing the test subjects. We also queried the test subjects to validate the explicability scores of the explicable plans generated using our solution approach.

Domain model

There are a lot of social norms followed by human drivers which are usually above explicit laws. This can include normative behavior while changing lanes or during turn-taking at intersections. As such this car domain has emerged as a vibrant testbed for research [82; 61] in the HRI/AI community in recent times. In this work, we explore the topic of explicable behavior of an autonomous car in this domain, especially as it related to mental modeling of the humans in the loop. In our autonomous car domain (modeled in PDDL), the autonomous car model $\mathcal{M}^R$ consists of lane and car objects as shown in Figure 3.3. The red car is the autonomous car in the experiments and all other cars are assumed to have human drivers. The car objects are associated with predicates defining the location of the car on a lane segment, status of left and right turn lights, whether the car is within the speed limit, the presence of a parked police car, and so on. The actions possible in the domain are with respect

to the autonomous car. These actions are *Accelerate*, *Decelerate*, *LeftSqueeze*, *RightSqueeze*, *LeftLight {On, Off}*, *RightLight {On, Off}*, *SlowDown* and *WaitAtStopSign*. To change a lane, three consecutive actions of *{Left, Right} Squeeze* are required.

From $\mathcal{M}^R$ we generated a total of 40 plans (consisting of both explicable and inexplicable behaviors) for 16 different planning problems. These plans were assessed by 20 human subjects, with each subject evaluating 8 plans (apart from 1 subject who evaluated 7 plans). Also, each plan was evaluated by 4 different subjects. The overall number of training samples was 159. The test subjects were required to have a state driving license. The subjects were provided with the initial state and goal of the car. After seeing the simulation the plan, they had to record whether they found each action explicable or not. The assessment had two parts: one part involved scoring each autonomous car action with 1, if explicable, and 0 otherwise (plan score was calculated as the fraction of actions in the plan that were labeled as explicable); the other part involved answering a questionnaire on the preconditions, effects of the robot actions. It consisted of 8 questions with *yes/no* answers. The questions used for constructing the domain and the corresponding answers are provided in Table 3.1. For each question, the answers with majority of votes were used by the domain modeler to construct $\mathcal{M}_h^R$. The questions with divided opinions (3 and 7) were not included in the models as some found that behavior explicable while some others did not. The $\mathcal{M}_h^R$ domain consists of the same state predicates but ended up with different action definitions with respect to preconditions, effects, and action-costs.

Following are two examples of how the feedback was interpreted by the domain modeler. For question 1, the majority of answers agreed with the statement. Therefore for actions Accelerate and SlowDown, additional preconditions like *(not (squeezingLeft ?x)), (not (squeezingLeft2 ?x)), (not (squeezingRight ?x)), (not (squeezingRight2 ?x))* were added, where x stands for car. For question 8, since the answers

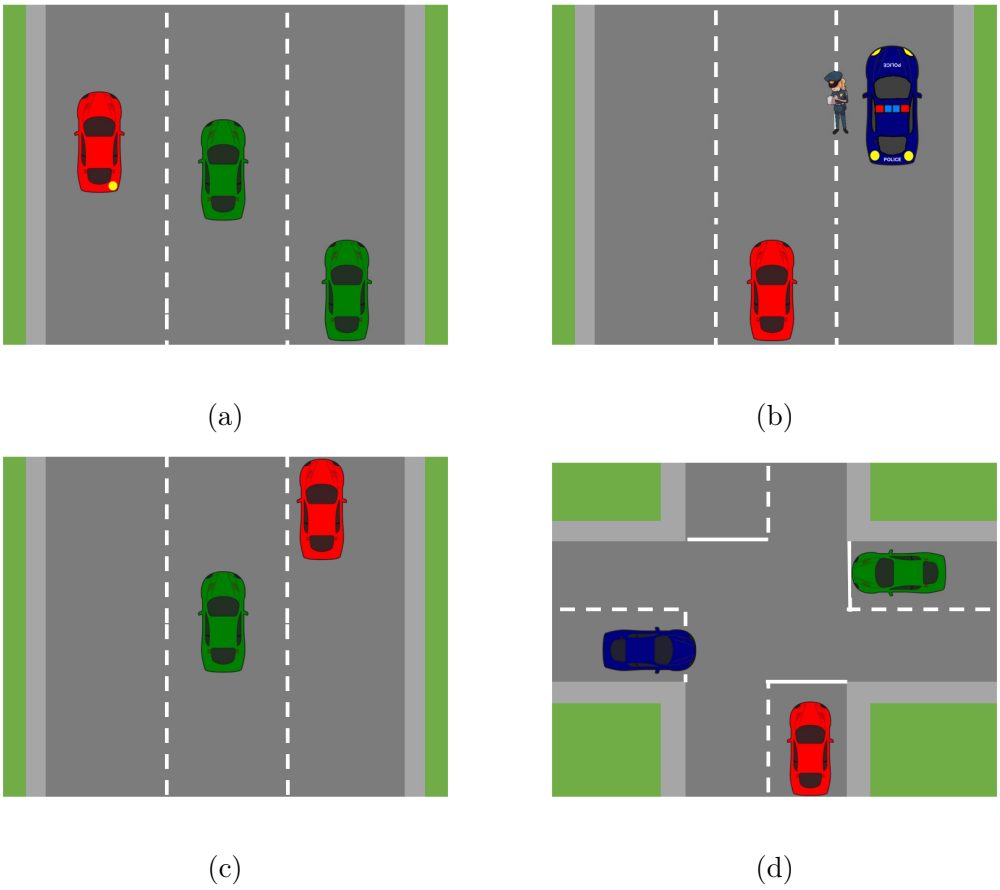


Figure 3.3: Simulated Autonomous Car Domain. Here Only the Red Car Is Autonomous. (a) the Autonomous Car Is Performing Lane Change Task (b) the Autonomous Car Is Performing a Move-over Maneuver (c) the Autonomous Car Is Trying to Merge to the Middle Lane and Is Confusing the Human Driver with Its Signals. (d) the Autonomous Car Is Waiting at a 4-way Stop Even Though It Is Its Turn to Cross.

No.	Questions	Yes	No
1	Autonomous car should always maintain itself in the center of the lane unless it is changing lanes	16	4
2	The car should automatically turn signal lights on before lane change.	20	0
3	The car should automatically turn signal lights on when the car starts swerving away from the center of the lane.	9	11
4	At a four-way stop, it should automatically provide turn signal lights when it is taking left or right turns.	18	2
5	It should slow down automatically when it is swerving off the center of the lane.	15	5
6	It should slow down automatically when there is an obstacle (pedestrian or parked vehicle) ahead in the lane.	17	3
7	Check one: When an emergency vehicle is parked on the rightmost lane, it should (1) automatically follow the move over maneuver, (2) whenever possible follow the move over maneuver.	9	11
8	Check one: At a four-way stop, it should (1) wait for the intersection to be clear and to be extra safe. (2) wait for the other cars unless it is its turn to cross over.	15	5

Table 3.1: The Questionnaire Used in the Human Study for Car Domain, and the Tally of Answers given by Participants. For the Last Two Questions, the Participants Were Asked to Choose One of the Two Options, and the “yes" Tally Corresponds to the First Answer, “no" To the Second.

agreed with choice 2, actions *waitAtStopSign1*, *waitAtStopSign2*, *waitAtStopSign3* were replaced by a new general action *waitAtStopSign*. This action removed predi-

Algorithm	Autonomous Car R^2 %	Delivery Robot R^2 %
Ridge Regression	53.66	31.42
AdaBoost Regression	61.31	51.27
Decision Tree Regression	74.79	39.61
Random Forest Regression	90.45	75.29

Table 3.2: Accuracy for Car and Delivery Domain.

cates *waiting1, waiting2, waiting3* from the action definition. Also, actions *atStopSignAccelerate*, *atStopSignLeft*, *atStopSignRight* were changed to remove the precondition *waiting3* (these actions thus had two definitions in $\mathcal{M}^R$ to allow for explicable behavior, one with higher cost).

Explicability Distance

For the training problems, explicable plans were generated using the model $\mathcal{M}_h^R$. Since some actions names were not common to both the domains, an explicit mapping was defined between the actions over the two domains. This mapping was done in order to support plan distance operations performed between plans in the two domains (for the plan distances to be used effectively common action names are required).

As noted in Definition 5, features of the regression model are the three plan distances and the target is the score associated with the plans. We tune the hyperparameters by performing grid search over parameters like the number of trees, depth of tree and the minimum number of nodes to perform sample split. The results for different learning models are as shown in Table 3.2. We tried several ensemble learning algorithms to improve the accuracy of our model, out of which random forest regression gave the best performance. Random forests allow selection of a random subset of features while splitting the decision node. We evaluate the goodness of fit

of the model, using the coefficient of determination or R^2. This value determines the measure by which the fitted model can explain the variations in the target values. This value lies between 0 to 1. Higher the R^2 value, better is the model fitted to the data. After training process, the new regression model was found to have 0.9045 R^2 value. That is to say, 90% of the variations in the features can be explained by our model. Our model predicts the explicability distance between the robot plans and human mental model plans, with a high accuracy.

Results

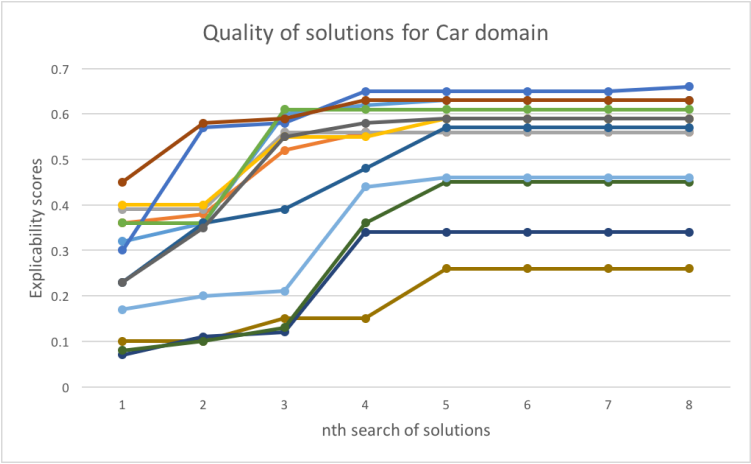


Figure 3.4: For the Car Domain Test Problems, the Graph Shows How the Search Process Finds Plans with Incrementally Better Explicable Scores. Each Color Line Represents One of the 13 Different Test Problems. The Markers on the Lines Represent a Plan Solution for That Problem. The Y-axis and the X-axis Represents the Explicability Scores of the Plans and the Solution Number Respectively.

We evaluated our approach on 13 different planning problems. We ran the algorithm with a high cost bound, in order to cover the most explicable candidate plans for all the problems. The Figure 3.4 reports the explicability scores of the first 8 solu-

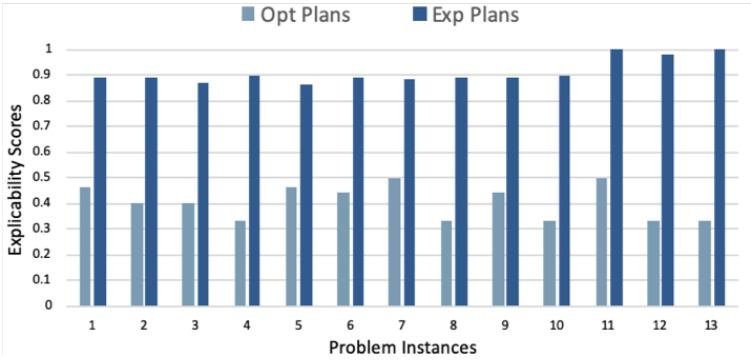


Figure 3.5: For the Car Domain, the Optimal and Explicable Plans Were Compared for Their Explicability Scores.

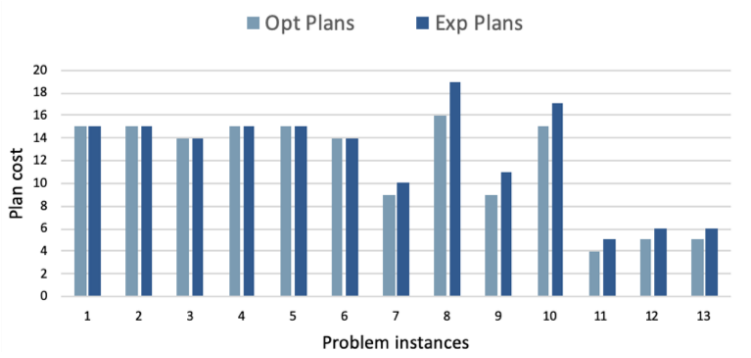


Figure 3.6: For the Car Domain, the Optimal and Explicable Plans Were Compared for Their Explicability Scores.

tions generated by our algorithm for 13 test problems. From this graph, we note that the reconciliation search is able to develop plans with incrementally better explicability scores. These are the internal explicability scores (produced by the explicability distance function).

The plan scores given by 10 test subjects were computed as the ratio of explicable actions in the plans to the total plan length. Testing phase protocol was same as that of the training phase, except for the questionnaire. The plan scores were averaged over the number of test subjects. In this domain, an example inexplicable plan for changing lanes from *l2* to *l1* would be *LeftSqueeze-l2-l1, LeftSqueeze-l2-l1,*

LeftSqueeze-l2-l1, LeftLightOn whereas the corresponding explicable plan would be *LeftLightOn, LeftSqueeze-l2-l1, LeftSqueeze-l2-l1, LeftSqueeze-l2-l1*. The ordering of *LeftLightOn* action decides whether the plan is explicable. From Figure 3.5, the test subjects provided higher explicability scores for the explicable plans than the optimal plans for all 13 problems. From Figure 3.6, we see for the last 7 problems, our planner generated explicable plans with a cost higher than that of optimal plans; a planner insensitive to explicability would not have been able to find these expensive but explicable plans. This additional cost can be seen as the price the robot pays to make its behavior explicable to the human. For the first 6 problems, even though the cost is same as that of the optimal plans, a planner insensitive to explicability cannot guarantee the generation of explicable plans.

3.3.4 Evaluation using Robot based Delivery Domain

This domain is designed to demonstrate inexplicable behaviors of a delivery robot. The robot can deliver parcels/electronic devices and serve beverages to the humans. It has access to three regions namely kitchen, reception and employee desk. For the evaluation, we used a Fetch robot, which has a single arm with parallel grippers for grasping objects. It delivers beverages, parcels, and devices using a tray. Whenever the robot carries the beverage cup there is some risk that the cup may tip over and spill the contents all over the electronic items on the tray. Here the robot has to learn the context of carrying devices and beverages separately even if it results in an expensive plan (in terms of cost or time) for it. A sample plan in this domain with explicable and inexplicable plans is illustrated in Figure 3.7. Here, in the inexplicable version, the robot delivers device and beverage together. Although it optimizes the plan cost, the robot may tip the beverage over the device. Whereas, in the explicable version robot delivers the device and beverage cup separately, resulting in an expensive

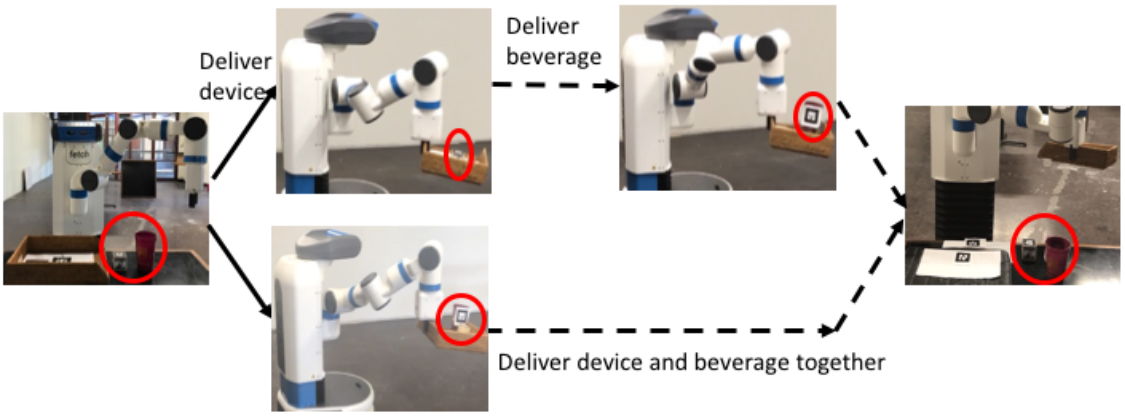
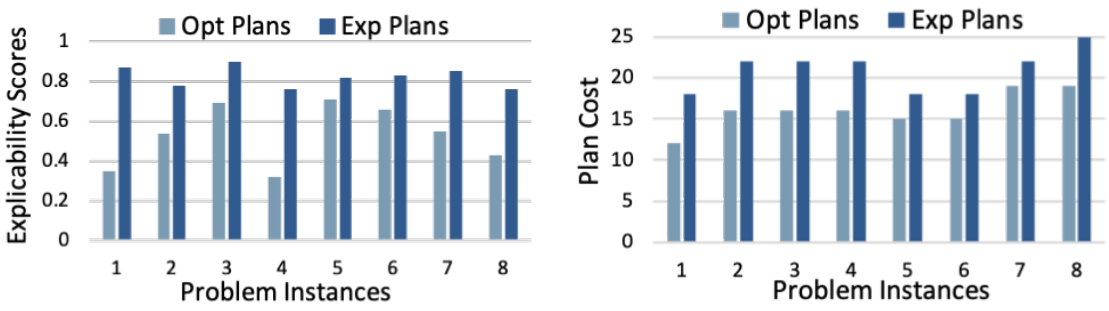


Figure 3.7: The Goal of the Robot Is to Deliver the Device and Beverage Cup to the Destination. In the Cost-optimal Plan, Robot Delivers Both the Items Together, Whereas in the Explicable Plan the Robot Delivers the Items Separately. A Video Demonstration Can Be Viewed at `https://bit.ly/2JweeYk`

plan due to multiple trips back and forth. A video demonstration can be viewed at `https://bit.ly/2JweeYk`.

Domain and explicability distance

This domain is also represented in PDDL. The robot model has the following actions available: `pickup`, `putdown`, `stack`, `unstack` and `move`. The domain modeler provided $\mathcal{M}_h^R$ based on usual expectations of robots with a similar form factor. For example, in $\mathcal{M}_h^R$, certain actions which could be perceived riskier (like, carrying the device and cup in the same tray) had a higher cost due to the possibility of damaging the items. Thus $\mathcal{M}_h^R$ incentivizes the planner to choose safer actions. Both models have same state space and action representation. The model differences lie in the action-costs as well as preconditions and effects of actions. There were 20 plan instances created for each of the 13 planning problems. Each of the plans was labeled by 2



(a) (b)

Figure 3.8: For Delivery Domain Test Problem Instances, the Optimal and Explicable Plans Were Compared for (a) Plan Costs (B) Explicability Scores Provided by Test Subjects.

human subjects, which resulted in 40 different training samples (some problems have multiple solution plans). The performance of different ensemble learning techniques is as shown in Table 3.2. We again use the random forest regression model with an accuracy of 75%.

Results

For evaluation of this domain, 8 new planning problems (similar to the one shown in Figure 3.7) were used. These plan instances with a pictorial representation of intermediate behavioral states were labeled by 9 test subjects. For testing phase, the same protocol was followed as that in training phase. In Figure 3.8a, we compare the explicability scores provided by test subjects. The explicability scores provided by the subjects are higher for explicable plans. Some plans involved the robot stacking cups over devices to generate cost-optimal plans. These plans ended up receiving least scores. Figure 3.8b shows the comparison between the plan costs. Whenever the items consist of beverage cups, the robot has to do multiple trips, therefore all

the explicable plans are more expensive than the optimal plans. For such scenarios, if a robot uses a cost-optimal planner, the plans chosen will always be inexplicable with respect to the plan context.

3.4 Model-Free Explicable Planning

If the robot does not have access to the human's mental model, then an approximation of it can be learned. This approach shows that it is not necessary to build a full-fledged planning model of the human's mental model, rather it is enough to predict the next explicable action given a plan prefix. In this setting, the human mental model is learned in the form of a labeling scheme used by the humans-in-the-loop. Here the underlying hypothesis is that the humans tend to associate abstract tasks or sub-goals to actions in a plan. If the human-in-the-loop is able to associate any domain-specific label to an action in a plan then that action is assumed to be explicable, otherwise, the action is considered inexplicable. Such a labeling scheme is learned using conditional random fields from training examples annotated by the humans. The learned model is used as a heuristic function in the planning process.

3.4.1 Problem Formulation

In this case, since the $\mathcal{M}_h^R$ is not known beforehand, the distance function $\delta_{\mathcal{M}_h^R}(\cdot,\cdot)$ in Definition 2 is approximated using a learning method. We postulate that the humans understand robot plans by associating abstract tasks with actions, which can be considered as a labeling process. Based on this, we assume that $\delta_{\mathcal{M}_h^R}(\pi_{\mathcal{M}^R}, \pi_{\mathcal{M}_h^R})$ can be functionally decomposed as: $\delta_{\mathcal{M}_h^R}(\pi_{\mathcal{M}^R}, \pi_{\mathcal{M}_h^R}) = F \circ \mathcal{L}^*(\pi_{\mathcal{M}^R})$, where F is a domain-independent function that takes plan labels as input, and $\mathcal{L}^*$ is the labeling scheme of the human for robot plans based on $\mathcal{M}_h^R$.

3.4.2 Labeling

Given the domain, we assume that a set of task labels T is provided to label robot actions: $T = \{T_1, T_2, \ldots, T_M\}$. In this formulation, we assume that explicability represents the association between abstract tasks and robot actions; each action in a plan is associated with an action label. The set of action labels for explicability is the power set of task labels: $\mathcal{L} = 2^T$. When an action label includes multiple task labels, the action is interpreted as contributing to multiple tasks; when an action label is an empty set, the action is interpreted as inexplicable. When a plan is labeled, we can compute its explicability measure based on its action labels in a domain-independent way.

More specifically, given a domain, the explicability $\theta_{\pi_{\mathcal{M}^R}}$ of a robot plan $\pi_{\mathcal{M}^R}$ is computed by a mapping, $F_\theta : \mathcal{L}_{\pi_{\mathcal{M}^R}} \rightarrow [0, 1]$ (with 1 being the most explicable). $\mathcal{L}_{\pi_{\mathcal{M}^R}}$ denotes the sequence of action labels for $\pi_{\mathcal{M}^R}$, and F_θ computes the ratio between the number of actions with non-empty action labels and the number of all actions.

3.4.3 Learning Approach

To formulate a learning method, we consider the sequence of labels as hidden variables. The plan that is executed by the robot (which also captures the state trajectory), as well as any cognitive cues that may be obtained (e.g., from sensing) during the plan execution, constitute the observations. The graphical model that we choose for our learning approach is conditional random field (CRF) [62] due to its ability to model sequential data. An alternative would be HMMs; however, CRFs have been shown to relax assumptions about the input and output sequence distributions and hence are more flexible. The distributions that are captured by CRFs have the following form where Z is a normalization factor: $p(x, y) = \frac{1}{Z} \prod_A (\phi(x_A, y_A))$.

In the equation above, x represents the sequence of observations, y represents the sequence of hidden variables, and $\phi(x_A, y_A)$ represents a factor that is related to a subgraph in the CRF model associated with variables x_A and y_A. In our context, x are the observations made during the execution of a plan; y are the action labels. Each factor is associated with a set of features that can be extracted during the plan execution.

Features for Learning

Given a robot plan, the set of features that we have access to is the plan and its associated state trajectory:

Plan Features Given the robot model (specified in PDDL), the set of plan features for a_i includes the action description and the state variables after executing the sequence of actions $\langle a_0, ..., a_i \rangle$ from the initial state. This information can be easily extracted given the model. We use a linear-chain CRF. Although linear-chain CRF is very limited in capturing history information, we show that it is sufficient to distinguish between behavioral patterns in our evaluation domains via a combination of the state and action information (i.e., plan features described above). Moreover, our formulation is easily extensible to more general classes of CRFs. Given a robot plan $\pi_{\mathcal{M}_h^R} = \langle a_0, a_1, a_2, \ldots \rangle$, each action is associated with a set of features. Hence, each training example is of the following form: $\langle (F_0, \mathcal{L}_0), (F_1, \mathcal{L}_1), (F_2, \mathcal{L}_2), \ldots, \rangle$, where $\mathcal{L}_i$ is the action label for explicability of a_i. F_i is the set of features for a_i.

3.4.4 Plan Generation

Given a set of training examples in the form of annotated plans, we can train the CRF model to learn the labeling scheme. We discuss two ways to use the learned

CRF model.

Plan Selection

The most straightforward method is to perform plan selection on a set of candidate plans which can simply be a set of plans that are within a certain cost bound of the optimal plan. Candidate plans can also be generated to be diverse with respect to various plan distances. For each plan, the robot must first extract the features of the actions as we discussed earlier. It then uses the trained model (denoted by $\mathcal{L}_{\mathcal{CRF}}$) to produce the labels for the actions in the plan. The explicability score of the plan can then be computed as discussed before. These scores can then be used to choose a plan that is more explicable.

Plan Synthesis

A more efficient way is to incorporate these measures as heuristics into the planning process. Here, we consider the FastForward (FF) planner with enforced hill-climbing [44]. To compute the heuristic value given a planning state, we use the relaxed planning graph to construct the remaining planning steps. However, since relaxed planning does not ensure a valid plan, we can only use action descriptions as plan features for actions that are beyond the current planning state when estimating the explicability scores. These estimates are then combined with the relaxed planning heuristic (which only considers plan cost) to guide the search.

3.5 Evaluation using Block Stacking Robot Domain

We evaluate our approach in a modified blocksworld domain with a physical robot Fetch. It simulates a smart manufacturing environment where robots are working alongside human coworkers. Although the human coworker and robot do not have

Algorithm 2 Model-Free Explicable Plan Synthesis

Input: $\mathcal{M}^R, \mathcal{L}_{\mathcal{CRF}}$

Output: $\pi^*_{\mathcal{P}_{Exp}}$

Push $\mathcal{I}^R$ into the open set O
while open set is not empty **do**
    s = GetNext(O)
    if G is reached **then**
        return s.plan (the plan that leads to s from $\mathcal{I}^R$)
    end if
    Compute all possible next states N from s
    for $n \in N$ **do**
        Compute the relaxed plan π_{RELAX} for n
        Concatenate s.plan (with plan features) with π_{RELAX} (with only action descriptions) as π^{prime}
        Compute and add other relevant features
        Compute $\mathcal{L}_{\pi^R} = \mathcal{L}_{\mathcal{CRF}}(\pi^{prime})$
        Compute $h = f(\theta, h_{cost})$ (f is a combination function; h_{cost} is the relaxed planning heuristic)
    end for
    if $h(n^*) < h^*$ ($n^* \in N$ with minimum h) **then**
        Clear O
        Push n^* into O; $h^* = h(n^*)$ (h^* is initially MAX)
    else
        Push all $n \in N$ into O
    end if
end while

common goals, synthesizing an explicable robot plan is quite crucial in this setting. As in a shared workspace, unexpected robot behaviors can lead to loss of trust or in worst case cause accidents. Therefore, to avoid such outcomes, the robot has to generate explicable plans. Here, we compare the plans generated by our approach against those by a cost-optimal planner (OPT) in terms of their explicability scores by performing user studies.

3.5.1 Domain Description

In this domain, the robot's goal (which is known to the human) is to build a tower of a certain height using blocks on the table. The towers to be built have different heights in different problems. There are two types of blocks, light ones, and heavy ones, which are indistinguishable externally but the robot can identify them based on the markers. Picking up the heavy blocks are more costly than the light blocks for the robot. Hence, the robot may sometimes choose seemingly more costly (i.e., longer) plans to build a tower from the human's perspective.

3.5.2 Experimental Setup

We generated a set of 23 problems in this domain in which towers of height 3 are to be built. The plans for these problems were manually generated and labeled as the training set. For 4 out of these 23 problems, the optimal plan is not the most explicable plan. To remove the influence of grounding, we also generated permutations of each plan using different object names for these 23 problems, which resulted in a total of about 15000 training samples. We then generated a set of 8 testing problems for building towers of various heights (from 3-5) to verify that our approach can generalize. Testing problems were generated only for cases where plans are more likely to be inexplicable. For each problem, we generated two plans, one using OPT

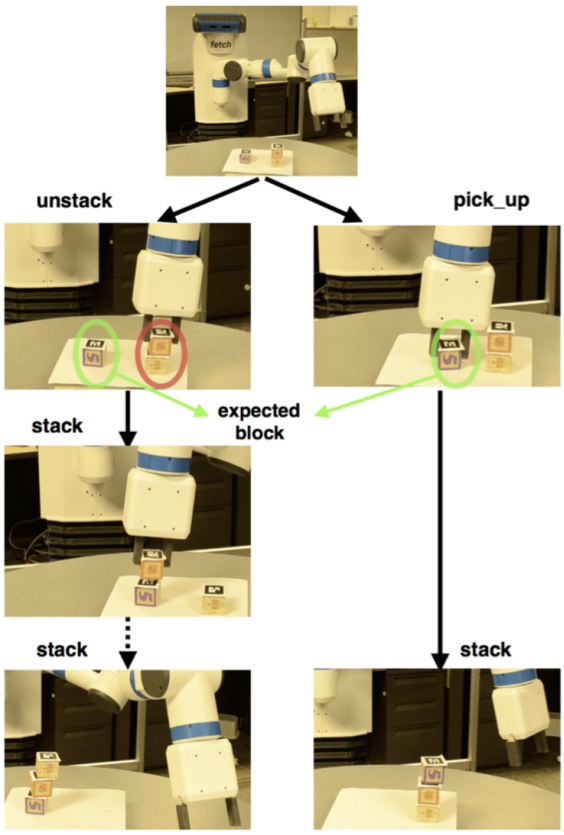


Figure 3.9: Execution of Two Plans Generated by Opt(Left) and Algorithm 2 (Right) for One out of the 8 Testing Scenarios. The Top Figure Shows the Setup Where the Goal Is to Build a Tower of Height 3. The Block Initially on the Left Side of the Table Is a Heavy Block. The Optimal Plan Involves Manipulating the Light Blocks (i.e., Putting the Two Light Blocks on Top of the Heavy One); The Explicable Plan Is More Costly since It Requires Moving the Heavy One. A Video of the Demonstration Can Be Viewed at `https://youtu.be/uSunoM628lw`.

and the other using Algorithm 2, and recorded the execution of these plans on the robot. We recruited 13 test subjects and each subject was tasked with labeling two plans (generated by OPT and our search algorithm respectively) for each of the 8

testing problems, using the recorded videos and following a process similar to that in the training phase. After labeling each plan, we also asked the subject to provide a score (1-10 with 10 being the most explicable) to describe how comprehensible the plan was overall.

3.5.3 Results

In this evaluation, we only use one task label "building tower". For all testing problems, the labeling process results in 77.8% explicable actions (i.e., actions with a task label) for OPT and 97.3% explicable actions for our search algorithm. The average explicability measures for our search algorithm and OPT are 0.98 and 0.78, and the average scores (out of 10) are 9.65 and 6.92, respectively. We analyze the results using a paired T-test which shows a significant difference between our search algorithm and OPT in terms of the explicability measures computed from the human labels and the overall scores ($p < 0.001$ for both). Furthermore, after normalizing the scores from the human subjects, the Cronbach's α value shows that the explicability measures and the scores are consistent for both our search algorithm and OPT (α = 0.78,0.67, respectively). These results verify that: 1) our explicability measure does capture the human's interpretation of the robot plans and 2) our approach can generate plans that are more explicable to humans. In Fig. 7, we present the plans for a testing scenario. The left part of the figure shows the plan generated by OPT and the right part shows the plan generated by our search algorithm. A video demonstration of the scenario can be viewed at `https://youtu.be/uSunoM628lw`

3.6 Concluding Remarks

In summary, we provided a solution to the problem of explicable planning in two different settings. In the first setting, we presented the problem of explicable plan

generation by modeling it in terms of plan distance measures studied in existing literature. We also demonstrated how a regression model on these distance measures can be learned from human feedback and used it to guide the robot's plan generation process to produce explicable plans. In the second setting, we showed how a labeling scheme can be learned to approximate the human mental model. We generated this labeling scheme by training a conditional random field model. We demonstrated the effectiveness of these approaches in simulated as well as physical robot domains.

Chapter 4

PLANNING FOR LEGIBLE BEHAVIOR

In this chapter, the discussion will focus on another type of an interpretable behavior, namely legibility. The notion of legibility allows the robot to implicitly communicate information about its goals, plans (or model, in general) to a human observer. For instance, consider a human robot cohabitation scenario consisting of a multi-tasking robot with varied capabilities that is capable of performing multitude of tasks in an environment. In such a scenario, it is crucial for the robot to aid the human's goal or plan recognition process, as the human observer may not always know the robot's intentions or objectives beforehand. Hence, in such cases, it may be useful for the robot to communicate (either explicitly or implicitly) information that the human is unaware of. Since, the better the human is at identifying the robot's goals or plans accurately, the better is the overall team performance. However, explicit communication of objectives might not always be suitable. For instance, the *what, when* and *how* of explicit communication may require additional thought. Further, several other aspects like cost of communication (in terms of resources or time), delay in communication (communications signals may take time to reach the human), feasibility of communication (broken or unavailable sensors), etc., may also need to be considered. In such cases, the robot can simply synthesize a behavior that implicitly communicates the necessary information to the human observer.

We will discuss the notion of legibility from the perspective of an offline setting where the observer has partial observability of the robot's actions. That is, the robot has to synthesize legible behavior in a setting where the human observer has access to the observations emitted from the entire execution trace of the robot but these

observations do not always reveal the robot's exact action or state to the human. As the human is trying to infer the robot's goals or plans, she operates in a belief space due to the partial observability of the robot's activities. The robot has to modulate the human's observability and choose actions such that the ambiguity over specific goals or plans is reduced. We refer to this as the controlled observability planning problem (COPP). In the upcoming sections, we will see how the COPP formulation can be used to synthesize goal legible behavior as well as plan legible behavior given an offline setting where the observer has partial observability of the robot's activities.

4.1 Related Work

In the motion planning and robotics community, legibility [28; 26; 29; 54] has been a well-studied topic. However, this has been mostly looked at from the motion planning perspective, and the focus has been on optimizing the motion trajectories to reveal the goal. Recently this notion has also been studied in the task planning community under the name of transparent planning [66]. We borrow this notion and generalize it using the COPP framework to provide legibility of goals and plans when the human observer has partial observability of the robot's activities.

The problem of goal legibility is directly related to the work on goal and plan recognition [78; 79; 31; 88; 47; 69; 77]. Traditional plan recognition systems have focused on techniques to decipher the goals or plans being pursued by the robot, irrespective of whether the robot is indifferent to the human observers or if it actively cooperative or adversarial towards the observer. The notion of legible behavior makes the inherent assumption that the robot is not indifferent to its observers and is actively aiding the human's process of plan or goal recognition.

In the recent years, the problem of goal recognition design (GRD) [47; 48; 49; 99; 98] has also received increasing attention. The GRD problem involves using

environment changing actions to simplify the problem of legibility. Such environment changing actions are a special class of actions that can change observability. If the set of actions available to the robot include environment changing actions, the COPP-based legible planning problem would automatically solve the problem of goal recognition design by selecting actions that maximize legibility.

4.2 Controlled Observability Planning Problem

In this framework, we consider two agents: a robot and an observer. The robot has full observability of its activities. However, the observer only has partial observability of the robot's activities. The observer is aware of the robot's planning model and receives observations emitted as a side effect of the robot's execution. This framework supports an offline setting, and therefore the observer only receives the observations after the robot has finished executing the entire plan.

In this setting, the robot has a set of candidate goals, inclusive of its true goal. The candidate goals are the set of possible goals that the robot may achieve in the given environment. The observer is aware of the robot's candidate goal set but is unaware of the robot's true goal. We now introduce a general COPP framework that will be used to define the goal legibility and plan legibility problems in the upcoming sections.

Definition 6. *A **controlled observability planning problem** is a tuple, $\mathcal{P}_{\mathcal{CO}} = \langle \mathcal{D}, \mathcal{G}, \Omega, \mathcal{O} \rangle$, where,*

- $\mathcal{D} = \langle \mathcal{F}, \mathcal{A}, \mathcal{I} \rangle$ *is the planning domain of the robot.*
- $\mathcal{G} = \{G_1 \cup G_2 \ldots \cup G_{n-1} \cup G_A\}$ *is a set of candidate goal conditions, each defined by subsets of fluent instantiations, where G_A is the true goal of the robot.*

- $\Omega = \{o_i | i = 1, \ldots, m\}$ *is a set of* m *observations that can be emitted as a result of the action taken and the state transition.*

- $\mathcal{O} : (\mathcal{A} \times \mathcal{S}) \rightarrow \Omega$ *is a many-to-one observation function which maps the action taken and the next state reached to an observation in* Ω*. That is to say, the observations are deterministic, each* $\langle a, s' \rangle$ *pair is associated with a single observation but multiple pairs can be mapped to the same observation.*

The observer has access to $\mathcal{P}_{\mathcal{CO}}$, but is unaware of the true goal of the robot. The observation function can be seen as a sensor model, as modeled in several prior works [35; 6; 50]. The observer receives the partial observations of the robot's execution based on the mapping of this sensor model. That is, for every action taken by the robot and an associated state transition, the observer receives an observation. However, a single observation may be consistent with multiple action-state pairs because of the many-to-one formulation of $\mathcal{O}$. Therefore, the observer operates in the belief space. The robot takes the belief space of the observer into account during its planning process, so as to modulate the observability of the observer.

4.2.1 Observer's Belief Space

The observer may use its observations of the robot's activity to maintain a *belief state*. A belief state is simply a set of possible states consistent with the observations. We use $\hat{s}$ as a notational aid to denote a state that is a member of the belief state.

Definition 7. *The* ***initial belief****,* b_0*, induced by observation,* o_0 *is defined as,* $b_0 = \{\hat{s}_0 \mid \mathcal{O}(\emptyset, s_0) = o_0 \wedge \mathcal{O}(\emptyset, \hat{s}_0) = o_0\}$.

Whenever a new action is taken by the robot, and the state transition occurs, the observer's belief updates as follows:

Definition 8. *A **belief update**, b_{i+1} for belief b_i is defined as,* $b_{i+1} = update(b_i, o_{i+1}) = \{\hat{s}_{i+1} \mid \exists \hat{a},\ \Gamma(\hat{s}_i, \hat{a}) \models \hat{s}_{i+1} \wedge \hat{s}_i \in b_i \wedge \mathcal{O}(\hat{a}, \hat{s}_{i+1}) = o_{i+1}\}$.

A sequence of belief updates gives us the observer's belief sequence that is consistent with the observation sequence emitted by the robot.

Definition 9. *A **belief sequence** induced by a plan p starting at state s_0, BS(p, s_0), is defined as a sequence of beliefs $\langle b_o, b_1, \ldots, b_n\rangle$ such that there exist $o_0, o_1, o_2, \ldots, o_n \in \Omega$ where,*

- $o_i = \mathcal{O}(a_i, s_i)$
- $b_{i+1} = update(b_i, o_{i+1})$

The set of plans that are consistent with the belief sequence of a given plan are called as belief plan set.

Definition 10. *A **belief plan set**, BPS(p, s_0) = $\{p_1, \ldots, p_n\}$, induced by a plan p starting at s_0, is a set of plans that are formed by causally consistent chaining of state sequences in $BS(p, s_0)$, i.e., BPS(p, s_0) = $\{\langle \hat{s}_0, \hat{a}_1, \hat{s}_1, \ldots, \hat{s}_n\rangle \mid \forall\ \hat{a}_j,\ \hat{s}_{j-1} \models pre(\hat{a}_j)\ \wedge\ \hat{s}_{j-1} \in b_{j-1}\ \wedge\ \hat{s}_j \models \hat{s}_{j-1} \cup add(\hat{a}_j) \setminus delete(\hat{a}_j)\ \wedge\ \hat{s}_j \in b_j\}$.*

The robot's objective is to generate a belief sequence in observer's belief space, such that, the last belief in the sequence satisfies certain desired conditions.

4.2.2 Complexity Analysis

In this section, we discuss the complexity results for $\mathcal{P}_{\mathcal{CO}}$. We prove that the plan existence problem for $\mathcal{P}_{\mathcal{CO}}$ is EXPSPACE-complete.

Theorem 1. *The plan existence problem for a controlled observability planning problem is EXPSPACE-hard.*

Proof. To show that the plan existence problem for $\mathcal{P}_{\mathcal{CO}}$ is EXPSPACE-hard, we will show that the NOD (No-Observability Deterministic) planning problem is reducible to $\mathcal{P}_{\mathcal{CO}}$. The plan existence problem for NOD has been shown to be EXPSPACE-complete [39; 81].

Let $\mathcal{P}_N = \langle \mathcal{F}_N, \mathcal{A}_N, \mathcal{I}_N, G_N, \mathcal{V} \rangle$ be a NOD planning problem, where, $\mathcal{F}_N$ is the set of fluents (or Boolean state variables), such that, state s is an instantiation of $\mathcal{F}_N$. $\mathcal{A}_N$ is a set of actions, such that, when an action $a \in \mathcal{A}_N$ is applied to a state, s_i, a deterministic transition to the next state occurs, $\Gamma(s_i, a) \models s_{i+1}$. $\mathcal{I}_N$ and $G_N = \{\phi_{G_N}\}$ are Boolean formulae that represent set of initial and goal states. $\mathcal{V} = \emptyset$ is the set of observable fluents. We are considering a NOD problem, therefore there are no observations. Since the underlying system state is unknown, the deterministic transition function does not reveal the hidden state. $\mathcal{P}_N$ can be expressed as a $\mathcal{P}_{\mathcal{CO}}$ problem, $\mathcal{P}_C = \langle \mathcal{D}_N, G_C, \Omega, \mathcal{O} \rangle$, where, $\mathcal{D}_N = \{\mathcal{F}_N, \mathcal{A}_N, \mathcal{I}_N\}$, such that $\mathcal{I}_N$ is a set of possible initial states, $G_C = \{\phi_{G_N}, \neg\phi_{G_N}\}$ is the set of goal states, $\Omega = \emptyset$ and $\mathcal{O} = \emptyset$. The goal set G_C consists of all states: the first goal formula is common with the NOD problem and the second one is the negation of it, which, in essence, encapsulates the rest of the states.

Suppose $\pi_{\mathcal{P}_C} = \langle a_1, \ldots, a_r \rangle$ is a 1-*legible* plan solution (see Definition 12) or 1-*ambiguous* plan solution (introduced in Chapter 6) to $\mathcal{P}_C$, such that, $\Gamma(\mathcal{I}_C, \pi_{\mathcal{P}_C}) \models \phi_{G_N}$ and the last belief $b_r \in BS(\pi_{\mathcal{P}_C}, \mathcal{I}_C)$ satisfies $|G \in G_C : \exists \hat{s} \in b_r, \hat{s} \models G| = 1$. Then according to the definition of $\mathcal{P}_N$, the plan $\pi_{\mathcal{P}_C}$ satisfies the following, $\exists \hat{s}_r \in b_r, \hat{s}_r \models \phi_{G_N}$ and therefore solves $\mathcal{P}_N$.

Conversely, suppose $\pi_{\mathcal{P}_N} = \langle a_1, \ldots, a_q \rangle$ is a plan solution to $\mathcal{P}_N$, such that, $\Gamma(\mathcal{I}_N, \pi_{\mathcal{P}_N}) \models G_N$. Let b_q be the belief associated with the last action in $\pi_{\mathcal{P}_N}$. Since it achieves the goal, we can say that $\exists \hat{s}_q \in b_q, \hat{s}_q \models \phi_{G_N}$. Given the problem of say, goal legibility, for legibility with respect to 1 goal, (i.e., for $j = 1$), b_q satisfies the

condition. A similar argument can be made for goal obfuscation with respect to 1 goal (i.e., $k = 1$), which is introduced in Chapter 6. Therefore $\pi_{\mathcal{P}_N}$ is a solution to $\mathcal{P}_C$. □

Theorem 2. *The plan existence problem for a controlled observability planning problem is EXPSPACE-complete.*

Proof. In $\mathcal{P}_{\mathcal{CO}}$, the planner operates in belief space and the search space is bounded by $2^{2^{|\mathcal{F}|}}$, where $|\mathcal{F}|$ is the cardinality of the fluents (or Boolean state variables). If there exists a plan solution for $\mathcal{P}_{\mathcal{CO}}$, it must be bounded by $2^{2^{|\mathcal{F}|}}$ in length. Any solution longer in length must have loops, which can be removed. Therefore, by selecting actions non-deterministically, the solution can be found in at most $2^{2^{|\mathcal{F}|}}$ steps. Hence, the plan existence problem for $\mathcal{P}_{\mathcal{CO}}$ is in NEXPSPACE. By Savitch's theorem [83], NEXPSPACE = EXPSPACE. Therefore, the plan existence problem for $\mathcal{P}_{\mathcal{CO}}$ is EXPSPACE-complete. □

4.2.3 Computing Solutions to COPP variants

Now we present a common algorithm template that can be used to compute plans for the COPP problem variants.

Algorithm for Plan Computation

In our algorithm, each search node is represented by an approximate belief estimate, which is an arbitrary Δ-sized subset of the belief state. A search node, b_Δ, consists of state s and a partial belief update, $\tilde{b}$ (cardinality of $\tilde{b}$ is given by $\Delta - 1$, where Δ is a counter). We borrow the concept of "width" from Geffner and Lipovetzky [36], but we consider the width in terms of cardinality of b_Δ. For example, when $\Delta = 1$, b_Δ only consists of the state s, when $\Delta = 2$, b_Δ consists of s, followed by a state from its

belief such that the augmented b_Δ has cardinality of 2. The successor node uses the s in b_Δ to generate the successor state, and b_Δ to update its partial belief. There are two loops in the algorithm: the outer and the inner loop. The outer loop maintains the cardinality of b_Δ by incrementing the value of Δ in each iteration, such that value of Δ ranges from $1, 2, \ldots, |\mathcal{S}|$. In the inner loop, a heuristic-guided forward search (for instance, greedy best first search) can be used to search over space of belief states of cardinality Δ. These loops ensure the complete exploration of the belief space.

Proposition 1. *The algorithm necessarily terminates in finite number of $|\mathcal{S}|$ iterations, such that, the following conditions hold:*
*(**Completeness**) The algorithm explores the complete solution space of $\mathcal{P}_{\mathcal{CO}}$, that is, if there exists a $\pi_{\mathcal{P}_{\mathcal{CO}}}$ that correctly solves $\mathcal{P}_{\mathcal{CO}}$, it will be found.*
*(**Soundness**) The plan, $\pi_{\mathcal{P}_{\mathcal{CO}}}$, found by the algorithm correctly solves $\mathcal{P}_{\mathcal{CO}}$ as ensured by the corresponding goal-test.*

The algorithm terminates either when a plan is found or after running the outer loop for $|\mathcal{S}|$ iterations. The outer loop ensures that all the paths in the search space are explored. The goal tests of the COPP problem variants ensure that the solutions are correct.

Optimization

In order to speed up the search process, we perform an optimization on the aforementioned algorithm. For each search node, b_Δ, apart from the approximate belief estimate, we maintain the full belief update b consistent with a path to s. The approximate belief update b_Δ can be generated by choosing Δ-sized combinations of states from the complete belief. For example, when $\Delta = 1$, b_Δ only consists of the state s but still maintains full belief update b, when $\Delta = 2$, b_Δ consists of a new

Algorithm 3 COOP Solution Plan Algorithm

Input: $\mathcal{P}_{\mathcal{CO}} = \langle \mathcal{D}, \mathcal{G}, \Omega, \mathcal{O} \rangle$ **Output:** Solution $\pi_{\mathcal{P}_{\mathcal{CO}}}$, observation sequence, $O_{\mathcal{P}_{\mathcal{CO}}}$

Initialize *open*, *closed*, *unopened* lists and the counter $\Delta \leftarrow 1$

$b_\Delta \leftarrow \{\mathcal{I}\}$; $b_0 \leftarrow \{\mathcal{O}(\emptyset, \mathcal{I})\}$ ▷ Initialize initial search node, initial belief

$open.push(\langle b_\Delta, b_0 \rangle, priority = 0)$

while $\Delta \leqslant |\mathcal{S}|$ **do**

  while $open \neq \emptyset$ **do**

    $b_\Delta, b, h(b_\Delta) \leftarrow open.pop()$

    if $|b_\Delta| \neq \Delta$ **then**

      $unopened.push(\langle b_\Delta, b \rangle, h(b_\Delta))$; **continue**

    end if

    $closed \leftarrow closed \cup b_\Delta$

    if $\langle b_\Delta, b \rangle \models$ GOAL-TEST$(\mathcal{G})$ **then**

      return $\pi_{\mathcal{P}_{\mathcal{CO}}}, O_{\mathcal{P}_{\mathcal{CO}}}$

    end if

    for $s' \in successors(s)$ **do**

      $o \leftarrow \mathcal{O}(a, s')$

      $b' \leftarrow$ Belief-Generation(b, o)

      $b'_\Delta = \langle s', \tilde{b}' \rangle$ ▷ $\tilde{b}'$ of size Δ-1

      $h(b'_\Delta) \leftarrow$ HEURISTIC-FUNCTION(b'_Δ, b')

      add b'_Δ to *open* if not in *closed*

    end for

  end while

  $\Delta \leftarrow \Delta + 1$

  copy items of *unopened* to *open*, empty *unopened*

end while

Algorithm 4 COOP Belief Update Procedure

1: **procedure** Belief-Generation(b, o)
2: $b' \leftarrow \{\}$
3: **for** $\hat{s} \in b$ **do**
4: **for** $\hat{a} \in \mathcal{A}$ **do**
5: **if** $\mathcal{O}(\hat{a}, \Gamma(\hat{s}, \hat{a})) = o$ **then**
6: $b' \leftarrow b' \cup \Gamma(\hat{s}, \hat{a})$
7: **end if**
8: **end for**
9: **end for**
10: **return** b'

combination of approximate belief of size 2 derived from the maintained full belief. When $\Delta = 1$, because of the check for duplicate states in the closed list, only one path to the search node is explored. Therefore, the use of Δ allows the search process to explore multiple paths leading to a particular search node. The complete b helps in finding the problem variant solutions faster at lower Δ values. We present the details of the optimization in Algorithm 3 and the belief generation procedure used by Algorithm 3 in Algorithm 4. In the following sections, we show how we customize the goal-test (line 11) and the heuristic function (line 18) to suit the needs of each of the COPP problem variants.

4.2.4 Variants of COPP

Within this framework, we will discuss the problem of goal legibility and plan legibility. With goal legibility, the objective of the robot is to convey *at most* j goals to the observer among the set of n candidate goals. With plan legibility, the set of candidate goals consists of only one goal, which is the robot's true goal. Even though

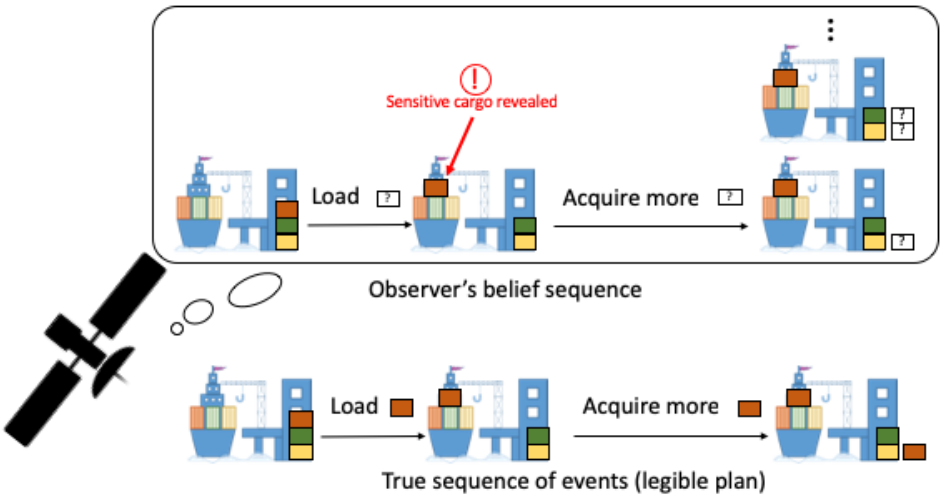
this goal is known to the observer, the objective of the robot with plan legibility is to convey to the observer *at least* m similar plans to the goal. We will see both of these problems in detail in the following sections. The COPP framework can be also used in adversarial environments. These variants will be covered in Chapter 6.

4.3 Goal Legibility

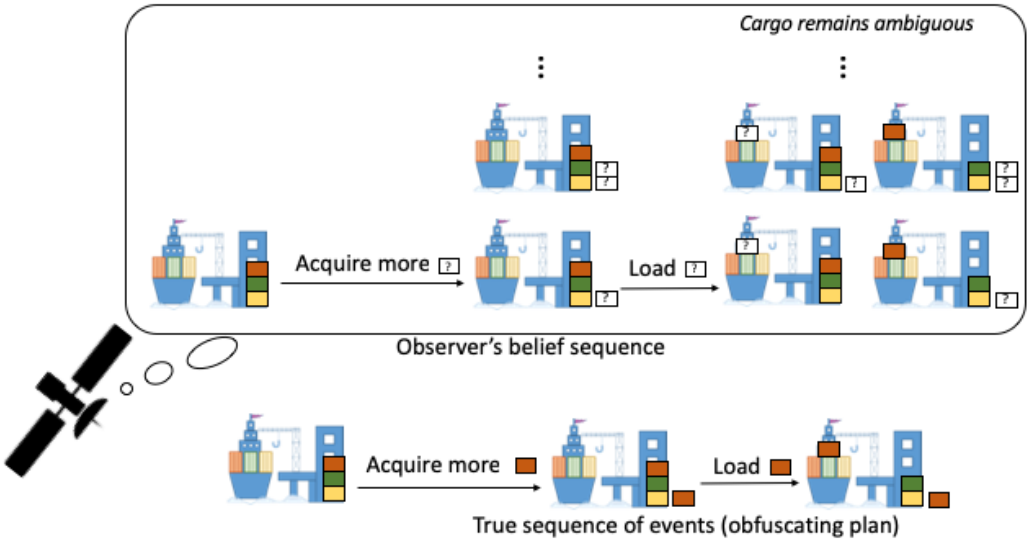
In this setting, the robot's objective is to convey some information about its candidate goal set to the human observer. This may involve communicating its true goal to the human, by ensuring at most one goal is consistent with the observation sequence produced. Or in general, the robot may want to communicate a set of at most j candidate goals inclusive of its true goal, to the human observer. Essentially, the point with goal legibility is to reduce the observer's ambiguity over the robot's possible goal set by narrowing it down to at most j goals.

Example

Let's understand this problem with an example. Consider a situation where the robot is a port management agent and the observer has sensors or informants at the port who provide partial information about the nature of activity being carried out at the port (refer Figure 4.1). For instance, when a specific crate is loaded onto the ship, the observer finds out that something was loaded, but not the identity of the loaded crate. The observer knows the initial inventory at the port, but when new cargo is acquired by the port, the observer's sensors reveal only that more cargo was received; they do not specify the numbers or identities of the received crates. A legible plan for loading the sensitive cargo (the red crate) and acquiring more cargo may first load the crate and then acquire more crates. This plan reveals the identity of the crate that was loaded based on the observers' information about the remaining cargo in



(a)



(b)

Figure 4.1: The Differences in Belief Sequences Induced by Different Plans for an Observer with Noisy Sensors.

the port: the final belief state has a unique crate loaded on the ship even though it retains uncertainty about the new cargo in the port. However, if the plan were to first acquire more cargo, the observer's sensors are insufficient to determine which crate was loaded: the plan maintains ambiguity in the observer's belief. This is reflected in the observer's belief state sequence, where the last belief state includes states with all different types of crates in the ship. Although both plans have the same cost and effects for the dock, one conveys the activity being carried out while the other adds more uncertainty. The COPP framework allows the robot to select plans that are legible in this manner.

Goal Legibility Planning Problem

In goal legibility problem, the robot's aim is to take actions exclusive to the goal so as to help the observer in goal deduction. Here we generalize the notion of goal legibility with respect to j number of goals.

Definition 11. *A **goal legibility planning problem** is a $\mathcal{P}_{CO}$, where, $\mathcal{G} = \{G_A \cup G_1 \cup \ldots \cup G_{n-1}\}$ is the set of n goals where G_A is the true goal of the robot, and $G_1, \ldots, G_{n-1}$ are the confounding goals.*

To solve this problem, the robot can generate a plan that conveys at most j candidate goals inclusive of its true goal. The robot has to ensure that the observation sequence of a legible plan is consistent with *at most* j goals so as to limit the number of goals in the observer's final belief state.

Definition 12. *A plan, π_j, is a **j-legible plan**, if $\Gamma(\mathcal{I}, \pi_j) \models G_A$ and the last belief, $b_n \in BS(\pi_j, \mathcal{I})$, satisfies the following, $|G \in \mathcal{G} : \exists s \in b_n, s \models G| \leqslant j$, where $1 \leqslant j \leqslant n$.*

Goal Diversity

Note that, the solutions to the goal legibility planning problems can be expressed in terms of a goal diversity measure in general. A goal diversity measure can be used to establish the extent of diversity among the set of goals present in the observer's belief. For instance, a simple goal diversity measure could be the cardinality of the goals satisfied in the last belief state, which is the measure admitted by Definition 12. However, other goal diversity measures can also be used to define the problems of goal legibility, and then *j-legible* solutions can be written as *at most j goal-diverse*.

4.3.1 Computing Goal Legible Plans

In the case of goal legibility, we run COPP search Algorithm 3 customized with the following goal test and heuristic function.

Goal test

In order to ensure that the computed plan is consistent with *at most* j true goals, we change our goal condition to additionally check whether at most $j-1$ confounding goals have been achieved in the observer's final belief, or it can be interpreted as at least $n-j$ goals are absent in the final belief.

Heuristic function

In this case, the robot's objective is to avoid at least $n-j$ goals, that is to be consistent with at most j goals. We achieve this by minimizing the heuristic cost to the true goal from the robot's actual state and to the $j-1$ confounding goals from the robot's belief state. However, we maximize the heuristic cost to other $n-j$ goals in order to

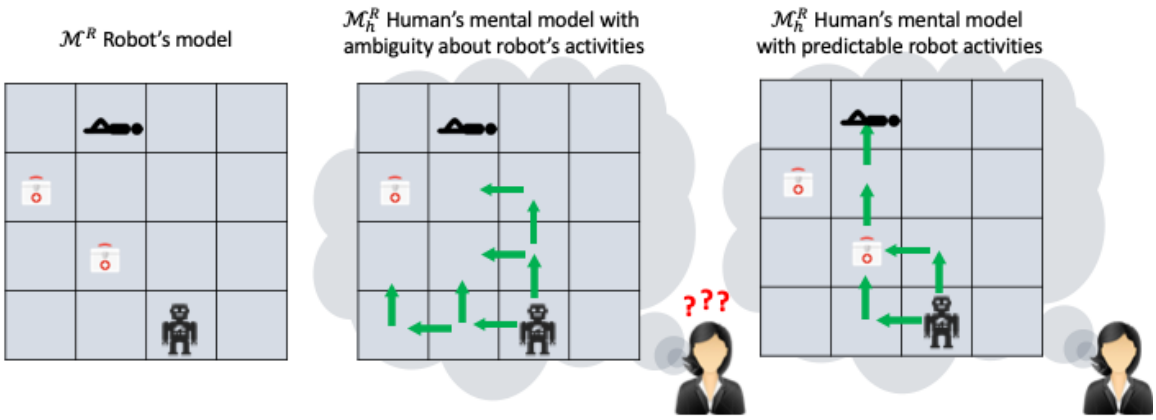


Figure 4.2: Illustration of the Impact of Plan Legibility on the Observer's Plan Inference Process.

achieve at most j goals in the last belief state. This is written as,

$$h(s) = h_{G_A}(s) + h_{\mathcal{G}_{j-1}}(b) - h_{\mathcal{G}_{n-j}}(b) \tag{4.1}$$

4.4 Plan Legibility

In this problem setting, the observer is aware of the robot's goal, since the candidate goal set consists of a single goal. Although, the observer is unaware of the robot's choice of plan to achieve that goal. In general, there can be multiple different plans that allow the robot to achieve a goal. This is further complicated by the fact that the observer has partial observability of the robot's activities. Therefore, in this problem setting the goal of the robot is to reduce the ambiguity over the possible plans to its goal. The robot can achieve this, by computing a plan whose observation sequence conforms to a set of plans that are similar in terms of their actions, thereby making it easy for the observer to guess the actions executed by the agent.

For instance, in Figure 4.2, the goal of the agent is to pick up a medkit and treat the victim on the disaster site. The observer has a noisy sensor model and it is not accurate enough to give the exact cell location of the robot. Therefore, the

observer relies on how many steps the robot has moved to guess the robot's path of execution. In the initial state shown in the leftmost subfigure in Figure 4.2, it can be seen that the robot has two choices of medkits to pick from. If it chooses the medkit as shown in the rightmost sub-figure, there are only two paths of length 2 towards the medkit and one path of length 2 towards the victim. In comparison, the middle sub-figure, shows the various different paths of length 4 that can lead to the other medkit, making it harder for the human to guess the robot's actions. Therefore, to reduce the ambiguity over its path, the robot chooses the medkit shown in the rightmost figure. This involves an observation sequence with two similar paths leading to the goal, making it easy for the human to guess some of the actions of the robot. Thus, the plan legibility problem can be solved by finding an observation sequence that is consistent with plans that are similar to each other.

Definition 13. *A **plan legibility planning problem** is a tuple,* $\mathcal{P}_{PL} = \langle \mathcal{D}, \mathcal{G}_{PL}, \Omega, \mathcal{O} \rangle$*, where,* $|\mathcal{G}_{PL}| = 1$.

4.4.1 Computing Plan Legible Plans

The solution to a plan legibility problem is an *m-similar* plan. An *m-similar* plan is a plan whose observation sequence is consistent with *at least* m similar plans to the goal, such that, these plans are *at most* d distance away from each other. In order to compute an *m-similar* plan, we need to keep track of the plans that are consistent with the observation sequence and reach the goal. To compute the diversity between all the pairs of plans consistent with the observation sequence, a plan distance measure [93; 74] like action distance, causal link distance, state sequence distance (introduced in Chapter 3) can be used. This approach can use any valid plan distance. We now define an *m-similar* plan.

Definition 14. *Two plans, p_1, p_2, are a* ***d-distant pair*** *with respect to distance function δ if, $\delta(p_1, p_2) = d$, where δ is a diversity measure.*

We use the belief plan set (BPS) introduced in Section 4.2 to define maximally d-distant pairs of plans associated with an observation sequence.

Definition 15. *A BPS induced by plan p starting at s_0 is* ***maximally d-distant****, $d_{max}(BPS(p, s_0))$, if $d = \max_{p1,p2 \in BPS(p,s_0)} \delta(p1, p2)$.*

Definition 16. *A plan, π_m, is an* ***m-similar plan****, if for a given value of d and distance function δ, $d_{max}(BPS(\pi_m, \mathcal{I})) \leq d$, $|BPS(\pi_m, \mathcal{I})| \geq m$, where $m \geq 2$ and every plan in $BPS(\pi_m, \mathcal{I})$ achieves the goal in $\mathcal{G}_{PL}$.*

In order to generate a solution to the plan legibility problem, we use the algorithm presented in Algorithm 3. Again, the goal test and heuristic function are customized to ensure that there are at least m similar plans to the true goal that are consistent with the observation sequence and the maximum distance between these plans is *at most* d.

Goal test

To ensure that the plans in BPS, induced by an *m-similar* plan, can achieve the goal in $\mathcal{G}_{PL}$, we check whether at least m plans are reaching the goal or not and whether the maximum distance between plans in BPS is at most d. Also in order to ensure termination of the algorithm, there is a cost-bound given as input to the algorithm.

Heuristic function

Apart from minimizing the heuristic cost to the goal, the customized heuristic given below also minimizes the d of $d_{max}(BPS(p, s_0))$ induced by plan p starting at s_0. This

decreases the maximum distance between the plan pairs in the BPS. This distance can be computed using a plan distance measure.

$$h(s) = h_{G_A}(s) + d_{max}(BPS(p, s_0)) \tag{4.2}$$

Plan Legibility as Offline Predictability

Plan legibility has been likened to the notion of offline predictability in a recent survey on different types of interpretable behaviors [15]. This is because plan legible behaviors allows the robot to reduce the observer's ambiguity over the possible plans that can be executed given a goal, which is also a property exhibited by predictable behaviors. However, predictable behaviors are also inherently easy to anticipate, either globally or locally. Globally predictable behavior [30] is a behavior that an observer would anticipate the robot to perform given a certain goal, versus locally predictable behavior [34] is a behavior where given a plan prefix, the rest of the suffix towards the given goal can be easily anticipated by the observer. This notion of predictability has been mostly explored in the motion planning community. However, plan legibility does not always lead to predictable behaviors. This is because in plan legibility, the emphasis is on making the robot's actions easy to guess given a corresponding observation sequence. However, the observation sequence in itself might not be globally or even locally predictable to the observer.

4.5 Empirical Evaluation of COPP Problem Variants

We now present an empirical evaluation of the solution approach presented in Section 4.2 to solve the goal legibility and plan legibility variants of COPP. Through evaluation, we intend to investigate the following objectives **(1)** comparison between run time and plan costs across the two problem variants versus the optimal solution to the true goal, **(2)** impact of Δ in Algorithm 3 for goal legibility.

Domain	Metrics	*j-leg*	*m-sim*	opt
Blocksworld	Time	213.68	100.96	0.99
	Length	12.4	11.28	9.6
Logistics	Time	247.9	45.05	8.31
	Length	27.25	27.94	23.74
Driverlog	Time	186.05	58.52	1.54
	Length	13.5	12.95	11.16

Table 4.1: Empirical Evaluation for the Two COPP Problem Variants Using the Optimization Presented in Algorithm 3 Versus the Optimal Plan Solution (Opt Column) to the True Goal. We Report the Average Time (in Seconds) and the Average Plan Length.

4.5.1 Domains and Experimental Setup

We use three IPC domains, namely `Blocksworld, Logistics` and `Driverlog` to evaluate our approach. For each of the domains, we randomly generated 50 problem instances. For the `Blocksworld` domain, we generated problems with 4 to 8 blocks and towers of maximum height 5 for both initial and goal states. After grounding, the smallest problem had 29 variables and 40 operators, and the largest problem had 89 variables and 144 operators. For the `Logistics` domain, we generated problems with goals consisting of 2 to 6 facts. After grounding, the smallest problem had 63 variables and 78 operators, and the largest problem had 63 variables and 198 operators. For the `Driverlog` domain, we generated problems with goals consisting of 2 to 6 facts. After grounding, the smallest problem had 33 variables and 100 operators, and the largest problem had 61 variables and 180 operators. We generated 5 random candidate goals (n=5) for each problem.

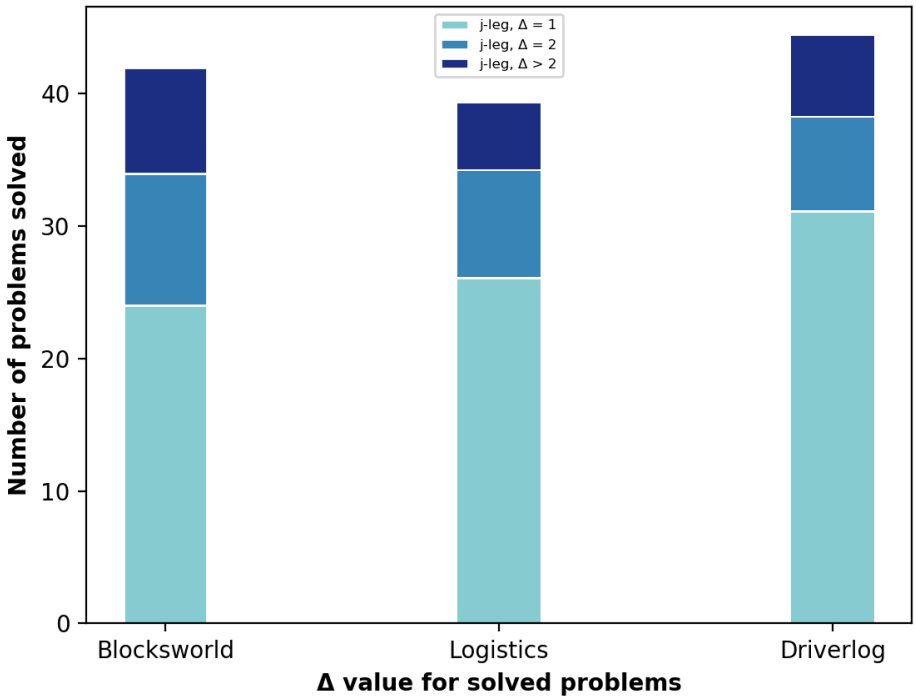


Figure 4.3: Empirical Evaluation of Δ in Algorithm 3 for Goal Legibility Variant. We Report the Number of Problem Instances Solved for Different Values of Δ.

The partially observable models have many-to-one mapping of action-state pairs to observation symbols. For the sake of simplicity, we used lifted action names as observation symbols. The grounded actions taken and associated states are mapped to the corresponding lifted action names. For the `Blocksworld` domain, the observation symbols were *pickup, putdown, stack, unstack*. For the `Logistics` domain, the observation symbols were *load-truck, unload-truck, load-airplane, unload-airplane, drive-truck, fly-airplane*. Finally, for the `Driverlog` domain, the symbols were *load-truck, unload-truck, board-truck, disembark-truck, drive-truck, walk*.

4.5.2 Results

The evalution results are presented in Table 4.1 and Figure 4.3. We modified the STRIPS planner Pyperplan [1] to implement our algorithms. We used the `hsa` [53]

heuristic of Pyperplan because it gave the best results in terms of computation time. We ran the experiments with $j = 2$, $m = 3$, and $d_{max} = 0.50$ for all the domains. We used action distance measure to compute the distance between plans. We ran our experiments on 12 core Intel Xeon CPU with an E5-2643 v3@3.40GHz processor with a 64G RAM with 20 minutes time-out. The following number of problems reached time-out before a solution could be found for the problem variants: 8/50 in the `Blocksworld`, 11/50 in `Logistics` and 6/50 in `Driverlog`. These problems were excluded from the results to ensure consistency across problems.

The Table 4.1 presents a comparison between the two different variants of the COPP framework. The *j-leg* solution takes slightly longer time. This is because, we have to ensure that at most j goals are present, which can be translated as at least $n - j$ goals are absent. For *j-leg*, we used 3 goals, where at most 2 goals were allowed, and at least 1 was disallowed to be present in the solution. For *m-sim*, the run time cost comes from maintaining BPS for each node. We also compare the time taken to compute an optimal plan to the true goal represented by the *opt* column. In general, both the COPP variants require comparatively longer run-time than the optimal plans due to the additional processing overhead resulting from modulating observer's beliefs. From Figure 4.3, we see that for goal legibility, most of the solutions are obtained for lower values of Δ, for all the 3 domains. That is, we can obtain solutions to the goal legibility problems in the first few iterations of the outer loop of Algorithm 3.

4.6 Concluding Remarks

We described the controlled observability planning framework in this chapter. Since the human observer has imperfect observations about the robot's activities, she operates in a belief state. Within this framework, we formulated the problems of goal

legibility and plan legibility. With goal legibility, the robot can convey at most j candidate goals (inclusive of its true goal) to the observer. While with plan legibility, the robot can make its activities legible by performing an *m-similar* plan whose observation sequence entails at least m plans that are at most d distance apart. We showed that these behaviors could be synthesized using a single algorithmic template. We evaluated the feasibility and performance of our approach using IPC domains.

Chapter 5

ENVIRONMENT DESIGN TO FACILITATE EXPLICABLE BEHAVIOR

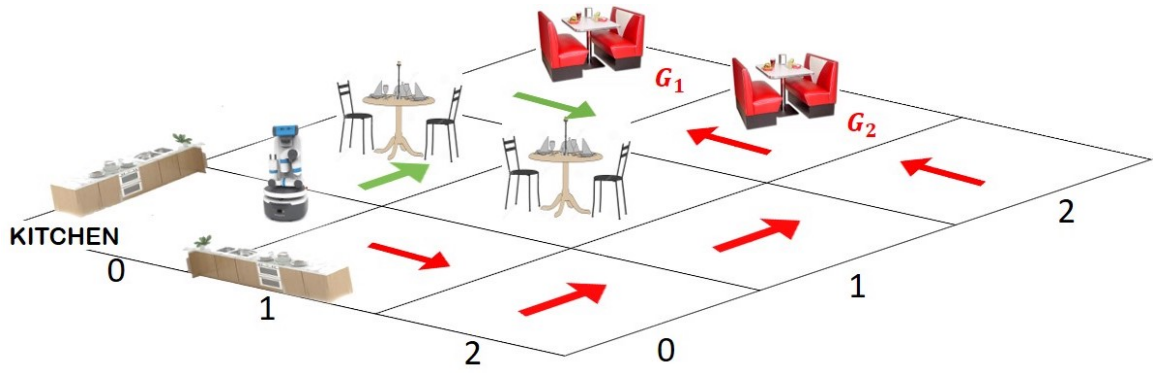
In this chapter, we will see how environment design approaches can be leveraged to make the environment, in which a robot and a human are interacting, more conducive to explicable robot behaviors. Whenever there are inconsistencies between the human's mental model of the robot and the robot's model, it should be able to reason over these inconsistencies to either generate *explicable behavior*, which is consistent with the human's expectations of its behavior [102; 57], or, *explain* its behavior with respect to the inconsistencies in the human's mental model [19; 92; 91]. However, the environment in which the robot is operating may not always be conducive to explicable behavior and/or to communicating explanations. As a result, making its behavior explicable may be prohibitively expensive for the robot. In addition, certain behaviors that are explicable with respect to the human's mental model may not be feasible for the robot.

Fortunately, in highly structured settings, where the robot is expected to solve repetitive tasks (like in warehouses, factory floors, restaurants, etc.), it might be feasible to *redesign the environment* in a way that improves explicability of the robot's behavior, given a set of tasks. This brings us to the notion of environment design which involves redesigning the environment to maximize (or minimize) some objective for the robot (like, optimal-cost to a goal, desired behavioral property) [101]. Thus, environment design can be used to boost the explicability of the robot's behavior, especially in settings that require solving repetitive tasks. Further, a one-time environment design cost to boost explicable behavior might be preferable over the repetitive cost overhead of explicable behavior borne by the robot. While the problem of

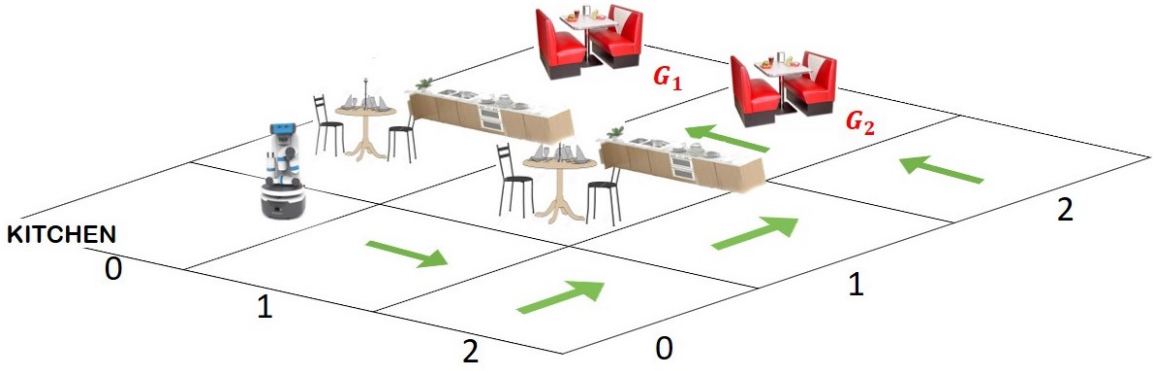
environment design for planning problems has been investigated under the umbrella of *goal and plan recognition design* [47; 72], they only form a subset of interpretable behaviors studied in the existing literature [14]. To the best of our knowledge, we are the first to explore the notion of environment design to maximize the explicability of a robot's behavior.

However, environment design alone may not be a panacea for explicability. For one, the design could be quite expensive, not only in terms of making the required environment changes but also in terms of limiting the capabilities of the robot. Moreover, in many cases, there may not be a single set of design modifications that will work for a given set of tasks. For instance, designing a robot with wheels for efficient navigation on the floor will not optimize the robot's motion up a stairwell. This means, to achieve truly effective synergy with autonomous robots in a shared space, we need a greater synthesis of environment design and human-aware behavior generation. This leads us to investigate a novel optimization space, that requires trading off one-time (but potentially expensive) design changes, against repetitive costs borne by the robot to exhibit explicable behavior.

In this work we propose a new design framework that balances the cost of modifying the environment with the cost of inexplicability of a robot's behavior given the human's mental model, and optimizes this objective given a set of tasks over a time horizon. Our work is the first to model the longitudinal aspect of explicable behavior, which captures the human's tolerance to inexplicability resulting from repetitive execution of tasks over a time horizon. While this has been an issue with existing formulations of explicable behavior, the longitudinal impact of inexplicability becomes especially critical in the context of environment design which affects agents more permanently. We leverage a planning compilation [90] to generate the most explicable plan for a task in a given environment and explore its theoretical properties. Through



(a) Explicable Behavior Is Costlier Without Design.



(b) Optimal Behavior Is Explicable with Design.

Figure 5.1: Use of Environment Design to Improve the Explicability of a Robot's Behavior in a Shared Environment.

empirical evaluation and demonstration of our approach in a simulated domain, we examine the properties of our optimization criterion and the various trade-offs that result from it.

Example

Consider a restaurant with a robot server (Figure 5.1a). Let G_1 and G_2 represent the robot's possible goals of serving the two booths: it travels between the kitchen and the two booths. The observers consist of customers at the restaurant. Given the position of the kitchen, the observers may have expectations on the route taken

by the robot. However, unbeknownst to the observers, the robot can not traverse between the two tables and can only take the route around the tables. Therefore, the path marked in red is the cheapest path for the robot but the observers expect the robot to take the path marked in green in Figure 5.1a.

In this environment, there is no way for the robot to behave as per the human's expectations. Applying environment design provides us with alternatives. For example, the designer could choose to build two barriers as shown in Figure 5.1b. With these barriers in place, the humans would expect the robot to follow the path highlighted in green. However, whether it is preferable to perform environment modifications or to bear the impact of inexplicable behavior depends on the cost of changing the environment versus the cost of inexplicability caused by the behavior.

In the upcoming sections, we will explore the details of this trade-off. We will also briefly discuss the formulation of the problem of environment redesign for other interpretable behaviors like legibility and predictability and how it is connected to the existing work on goal recognition design and plan recognition design respectively.

5.1 Related Work

This work explores the connection between two parallel threads of active research: one on environment design and the other on explicable behavior (introduced in Chapter 3). The problem of environment design is connected to that of mechanism design [73], which has been thoroughly investigated by the game theory community. Environment design [101] involves modifying the environment so as to maximize or minimize some objective for an agent [52]. The problem of design has been leveraged to simplify related problems like goal recognition [47], plan recognition [72], etc. These works have studied the possibility of modifying the environment so as to make the robot's behavior easily recognizable. The design modifications can be applied to

a robot's model in the form of action conditioning [51] or action pruning [47]. Such design strategies limit a certain set of actions from being applied either because of conditioning on other actions or because of being pruned altogether. Similarly, design modification in the form of sensor refinement [51] and sensor placement [50] can be applied to the observer's model, so as to improve the observer's recognition of the robot's goal. The problem of environment design has also been studied for stochastic actions [99; 98].

5.2 Background

We consider two agents: a robot and a human observer. In this section, we introduce the notion of inexplicability score and the problem of environment design with respect to these two agents.

Inexplicability Score

$\mathcal{M}^R = \langle \mathcal{F}, \mathcal{A}^R, \mathcal{I}^R, \mathcal{G}^R, c^R \rangle$ is the robot's model captured as a planning problem. The need for generating explicable behavior arises because the robot's planning model is different from the human's mental model of it. The difference can be in terms of a set of actions, the initial state or goal of the robot. Thus an explicable planning problem is $\mathcal{P}_{Exp} = \langle \mathcal{M}^R, \mathcal{M}_h^R, \delta_{\mathcal{M}_h^R} \rangle$, where $\mathcal{M}_h^R = \langle \mathcal{F}, \mathcal{A}_h^R, \mathcal{I}_h^R, \mathcal{G}_h^R, c_h^R \rangle$ represents the human's mental model of the robot model, and $\delta_{\mathcal{M}_h^R}$ is a distance function used by the human to compute the explicability of a plan. We assume the human mental model is given as an input. This is usually the case when any product is deployed and developers capture a generic user model which can be learned from prior interactions. In this work, we only focus on the reasoning aspects once we have the model, rather than focusing on the acquisition of such a model. $\Pi^*_{\mathcal{M}_h^R}$ represents the set of expected plans with respect to $\mathcal{M}_h^R$. A valid plan that solves $\mathcal{M}^R$ can exist anywhere on the

spectrum of inexplicability from high to low.

Definition 17. *The **inexplicability score**, $\mathcal{IE}(\cdot,\cdot,\cdot)$, of the robot's plan π^R that solves $\mathcal{M}^R$ is defined as follows for the human's mental model $\mathcal{M}_h^R$ and a distance function $\delta_{\mathcal{M}_h^R}(\cdot,\cdot)$:*

$$\mathcal{IE}(\pi^R, \mathcal{M}_h^R, \delta_{\mathcal{M}_h^R}) = \min_{\pi_h^R \in \Pi^*_{\mathcal{M}_h^R}} \delta_{\mathcal{M}_h^R}(\pi^R, \pi_h^R) \tag{5.1}$$

where $\delta_{\mathcal{M}_h^R}(\cdot,\cdot)$ is a distance function that assesses the difference between the two plans π^R and π_h^R.

The robot's objective is to minimize the inexplicability score in the human's mental model. We will use the notation $\Pi^*_{\mathcal{IE}(\cdot,\mathcal{M}_h^R,\delta_{\mathcal{M}_h^R})}$ (in the absence of the parameter π^R) to refer to the set of plans in the robot's model with the lowest inexplicability score, and $\mathcal{IE}_{min}(\mathcal{P}_{Exp})$ to represent the lowest inexplicability score associated with the set. Further, let f_{Exp} be the decision function used by the explicable robot: $f_{Exp}(\mathcal{P}_{Exp})$ represents the cheapest plan that minimizes the inexplicability score, i.e. $f_{Exp}(\mathcal{P}_{Exp}) \in \Pi^*_{\mathcal{IE}(\cdot,\mathcal{M}_h^R,\delta_{\mathcal{M}_h^R})}$ and $\neg\exists \pi' : \pi' \in \Pi^*_{\mathcal{IE}(\cdot,\mathcal{M}_h^R,\delta_{\mathcal{M}_h^R})}$ such that $c^R(\pi') < c^R(f_{Exp}(\mathcal{P}_{Exp}))$.

5.2.1 Environment Design

An environment design problem [101] takes as input the initial environment configuration along with a set of available modifications and computes a subset of modifications that can be applied to the initial environment to derive a new environment in which a desired objective is optimized.

We consider $\mathcal{M}^{R^0} = \langle \mathcal{F}^0, \mathcal{A}^{R^0}, \mathcal{I}^{R^0}, \mathcal{G}^{R^0}, c^{R^0} \rangle$ as the initial environment and ρ^R as the set of valid configurations of that environment: $\mathcal{M}^{R^0} \in \rho^R$. Let $\mathcal{O}$ be an arbitrary metric that needs to be optimized with environment design, i.e a planning

model with lower value for $\mathcal{O}$ is preferred. A design problem (adapted from [101]) is a tuple $\langle \mathcal{M}^{R^0}, \Delta, \Lambda^R, C, \mathcal{O} \rangle$ where, Δ is the set of all modifications, $\Lambda^R : \rho^R \times 2^\Delta \rightarrow \rho^R$ is the model transition function that specifies the resulting model after applying a subset of modifications to the existing model, $C : \Delta \rightarrow \mathbb{R}$ is the cost function that maps each design choice to its cost. The modifications are independent of each other and their costs are additive. We will overload the notation and use C as the cost function for a subset of modifications as well, i.e. $C(\xi) = \sum_{\xi_i \in \xi} C(\xi)$.

The set of possible modifications could include modifications to the state space, action preconditions, action effects, action costs, initial state and goal. In general, the space of design modifications, which are an input to our system, may also involve modifications to the robot itself (since the robot is part of the environment that is being modified). An optimal solution to a design problem identifies the subset of design modifications, ξ, that minimizes the following objective consisting of the cost of modifications and the metric $\mathcal{O}$: $\min \mathcal{O}(\Lambda^R(\mathcal{M}^{R^0}, \xi)),\ C(\xi)$.

5.3 Design for Explicability

In this framework, we not only discuss the problem of environment design with respect to explicability but also in the context of **(1)** a set of tasks that the robot has to perform in the environment, and **(2)** over the lifetime of the tasks i.e. the time horizon over which the robot is expected to repeat the execution of the given set of tasks. These considerations add an additional dimension to the environment design problem since the design will have lasting effects on the robot's behavior. In the following, we will first introduce the design problem for a single explicable planning problem, then extend it to a set of explicable planning problems and lastly extend it over a time horizon.

5.3.1 Design for a Single Explicable Problem

In the design problem for explicability, the inexplicability score becomes the metric that we want to optimize for. That is we want to find an environment design such that the inexplicability score is reduced in the new environment. This problem can be defined as follows:

Definition 18. *The design problem for explicability is a tuple,* $\mathcal{DP}_{Exp} = \langle \mathcal{P}^0_{Exp}, \Delta, \Lambda_{Exp}, C, \mathcal{IE}_{min} \rangle$*, where:*

- $\mathcal{P}^0_{Exp} \in \rho_{Exp}$ *is the initial configuration of the explicable planning problem, where* ρ_{Exp} *represents the set of valid configurations for* $\mathcal{P}_{Exp}$*.*
- Δ *is the set of available design modifications. The space of all possible modifications is the power-set* 2^{Δ}*.*
- $\Lambda_{Exp} : \rho_{Exp} \times 2^{\Delta} \rightarrow \rho_{Exp}$ *is the transition function over the explicable planning problem, which gives an updated problem after applying the modifications.*
- C *is the additive cost associated with each design in* Δ*.*
- $\mathcal{IE}_{min} : \rho_{Exp} \rightarrow \mathbb{R}^+$ *is the minimum possible inexplicability score in a configuration, i.e. the inexplicability score associated with the most explicable plan.*

With respect to our motivating example in Figure 5.1a, $\mathcal{DP}_{Exp}$ is the problem of designing the environment to improve the robot's explicability given its task of serving every new customer at a booth (say G_1) only once. The optimal solution to $\mathcal{DP}_{Exp}$ involves finding a configuration which minimizes the minimum inexplicability score. We also need to take into account an additional optimization metric which is the effect of design modifications on the robot's plan cost. That is, we need to examine to what extent the decrease in inexplicability is coming at the robot's expense. For

instance, if you confine the robot to a cage so that it cannot move, its behavior becomes completely and trivially explicable, but the cost of achieving its goals goes to infinity.

Definition 19. *An **optimal solution** to $\mathcal{DP}_{Exp}$, is a subset of modifications ξ^* that minimizes the following:*

$$\min\ \mathcal{IE}_{min}(\ \mathcal{P}^*_{Exp}),\ C(\xi^*), c^R(f_{Exp}(\mathcal{P}^*_{Exp})) \tag{5.2}$$

where $\mathcal{P}^*_{Exp} = \Lambda_{Exp}(\mathcal{P}^0_{Exp}, \xi^*)$ is the final modified explicable planning problem, $\mathcal{IE}_{min}(\cdot)$ represents the minimum possible inexplicability score for a given configuration, $C(\xi^*)$ denotes the cost of the design modifications and $c^R(f_{Exp}(\mathcal{P}^*_{Exp}))$ is the cost of the cheapest most explicable plan in a configuration.

5.3.2 Design for Multiple Explicable Problems

We will now show how $\mathcal{DP}_{Exp}$ evolves when there are multiple explicable planning problems in the environment that the robot needs to solve. When there are multiple tasks there may not exist a single set of design modifications that may benefit all the tasks. In such cases, a solution might involve performing design modifications that benefit some subset of the tasks while allowing the robot to act explicably with respect to the remaining set of tasks. Let there be k explicable planning problems, given by the set $\mathbf{P}_{Exp} = \{\langle \mathcal{M}^R(0), \mathcal{M}^R_h(0), \delta_{\mathcal{M}^R_h(0)}\rangle, \ldots, \langle \mathcal{M}^R(k), \mathcal{M}^R_h(k), \delta_{\mathcal{M}^R_h(k)}\rangle\}$, with a categorical probability distribution $\mathcal{D}$ over the problems. We use $\mathcal{P}_{Exp}(i) \in \mathbf{P}_{Exp}$ to denote the i^{th} explicable planning problem. These k explicable problems may differ in terms of their initial state and goal conditions. Now the design problem can be defined as:

$$\mathcal{DP}_{Exp,\mathcal{D}} = \langle \mathbf{P}^0_{Exp}, \mathcal{D}, \Delta, \Lambda_{Exp}, C, \mathcal{IE}_{min,\mathcal{D}}\rangle, \tag{5.3}$$

where $\mathbf{P}^0_{Exp}$, is the set of planning tasks in the initial environment configuration, $\mathcal{IE}_{min,\mathcal{D}}$ is a function that computes the minimum possible inexplicability score in a given environment configuration by taking the expectation over the minimum inexplicability score for each explicable planning problem, i.e., $\mathcal{IE}_{min,\mathcal{D}}(\mathbf{P}_{Exp}) = \mathbb{E}[\mathcal{IE}_{min}(\mathcal{P}_{Exp})]$, where $\mathcal{P}_{Exp} \sim \mathcal{D}$. With respect to our running example, $\mathcal{DP}_{Exp,\mathcal{D}}$ is the problem of designing the environment given the robot's task of serving every new customer only once at either of the booths (G_1, G_2) with probability given by $\mathcal{D}$.

The solution to $\mathcal{DP}_{Exp,\mathcal{D}}$ has to take into account the distribution over the set of explicable planning problems. Therefore the optimal solution is given by:

$$\min \quad \mathcal{IE}_{min,\mathcal{D}}(\ \mathbf{P}^*_{Exp}),\ C(\xi^*),\ \mathbb{E}[c^R(f_{Exp}(\mathcal{P}^*_{Exp}))] \tag{5.4}$$

where $\mathcal{P}^*_{Exp} \sim \mathcal{D}$. A valid configuration minimizes the minimum possible inexplicability score, which involves 1) expectation over minimum inexplicability scores for each explicable planning problem; 2) the cost of the design modifications (these modifications are applied to each explicable planning problem); and 3) the expectation over the cheapest most explicable plan for each explicable planning problem.

5.3.3 Longitudinal Impact on Explicable Behavior

The process of applying design modifications to an environment makes more sense if the tasks are going to be performed repeatedly in the presence of a human (i.e. the robot does not have to bear the cost of being explicable repeatedly). This has quite a different temporal characteristic in comparison to that of execution of one-time explicable behavior. For instance, design changes are associated with a one-time cost (i.e. the cost of applying those changes in the environment). On the other hand, if we are relying on the robot to execute explicable plans at the cost of foregoing optimal plans, then it needs to bear this cost multiple times in the presence of a human over

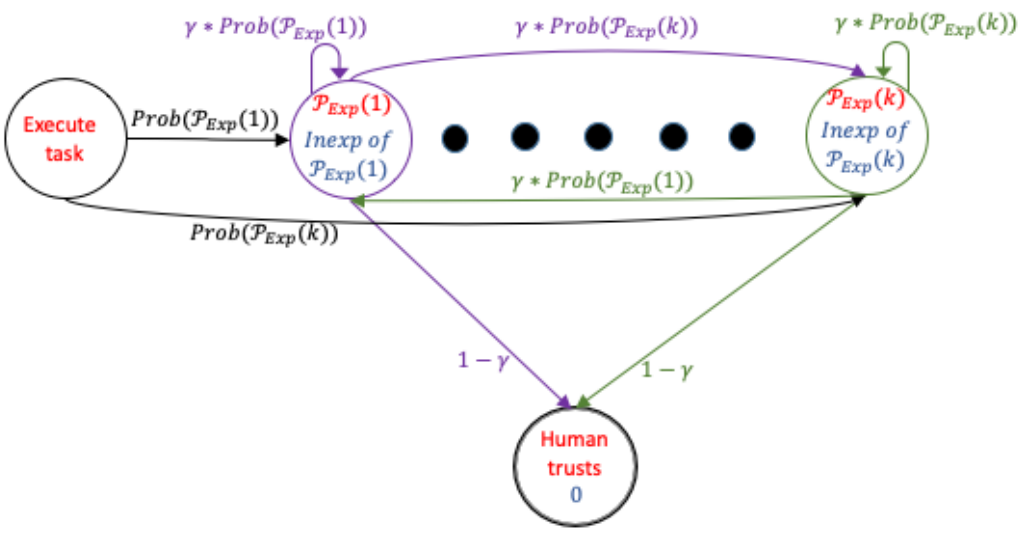


Figure 5.2: Illustration of Longitudinal Impact on Explicability. *Prob* Determines the Probability Associated with Executing Each Task in $\mathbf{P}_{Exp}$. For Each Task, the Reward Is Determined by the Inexplicability Score of That Task. The Probability of Achieving This Reward Is Determined by $\gamma \times$ Probability of Executing That Task. Additionally, with a Probability $(1-\gamma)$ the Human Ignores the Inexplicability of a Task and the Associated Reward Is given by an Inexplicability Score Of 0.

the time horizon.

We will use a discrete time formulation where the design problem is associated with a time horizon $\mathcal{T}$. At each time step, one of the k explicable planning problems is chosen. Now the design problem can be defined as:

$$\mathcal{DP}_{Exp,\mathcal{D},\mathcal{T}} = \langle \mathbf{P}^0_{Exp}, \mathcal{D}, \Delta, \Lambda_{Exp}, C, \mathcal{IE}_{min,\mathcal{D}}, \mathcal{T} \rangle \tag{5.5}$$

In our running example, $\mathcal{DP}_{Exp,\mathcal{D},\mathcal{T}}$ is the problem of designing the environment given the robot's task of serving the same customer at either of the booths with a distribution $\mathcal{D}$ over a horizon $\mathcal{T}$.

In the past literature, the explicable behavior has been studied with respect to a single interaction with a human over a given task [102; 57]. However, we consider

a time horizon, $\mathcal{T} > 1$, over which the robot's interaction with the human may be repeated multiple times for the same task. This means the human's expectations about the task can evolve over time. This may not be a problem if the robot's behavior aligns perfectly with the human's expectations. Although, if the robot's plan for a given task is associated with a non-zero inexplicability score, then the human is likely to be more surprised the very first time she notices the inexplicable behavior than she would be if she noticed the inexplicable behavior subsequent times. As the task is performed over and over, the amount of surprise associated with the inexplicable behavior starts decreasing. In fact, there is a probability that the human may ignore the inexplicability of the robot's behavior after sufficient repetitions of the task. We incorporate this intuition by using discounting.

Figure 5.2 illustrates the Markov reward process to represent the dynamics of this system. Let $(1 - \gamma)$ denote the probability that the human will ignore the inexplicability of the robot's plan, i.e, the reward will have inexplicability score 0. γ times the probability of executing a task represents the probability that the reward will have the minimum inexplicability score associated with that task. Assuming $\gamma < 1$, the minimum possible inexplicability score for a set of explicable planning problems is:

$$\begin{aligned} f_{\mathcal{T}}(\mathcal{IE}_{min,\mathcal{D}}(\mathbf{P}_{Exp})) &= \mathcal{IE}_{min,\mathcal{D}}(\mathbf{P}_{Exp}) \\ &\quad + \gamma * \mathcal{IE}_{min,\mathcal{D}}(\mathbf{P}_{Exp}) + \ldots + \\ &\quad \gamma^{T-1} * \mathcal{IE}_{min,\mathcal{D}}(\mathbf{P}_{Exp}) \\ f_{\mathcal{T}}(\mathcal{IE}_{min,\mathcal{D}}(\mathbf{P}_{Exp})) &= \frac{1-\gamma^{\mathcal{T}}}{1-\gamma} * \mathcal{IE}_{min,\mathcal{D}}(\mathbf{P}_{Exp}) \end{aligned} \tag{5.6}$$

Thus the optimal solution to $\mathcal{DP}_{Exp,\mathcal{D},\mathcal{T}}$ is given by:

$$\begin{aligned} \min \quad & f_{\mathcal{T}}(\mathcal{IE}_{min,\mathcal{D}}(\mathbf{P}^*_{Exp})),\ C(\xi^*), \\ & \mathbb{E}[c^R(f_{Exp}(\mathcal{P}^*_{Exp}))] * \mathcal{T} \end{aligned} \tag{5.7}$$

where, $\mathcal{P}^*_{Exp} \sim \mathcal{D}$. The optimal solution is a valid configuration that minimizes 1) the minimum possible inexplicability over the set of explicable planning problems given the human's tolerance to inexplicable behavior; 2) one-time cost of the design modifications; and 3) the expectation over the cheapest most explicable plan for each explicable planning problem given a time horizon. Note that, since the design cost is not discounted and we always make the design changes before the task is solved, there is never a reason to delay the design execution to future steps in the horizon. Instead it can be executed before the first time step.

5.4 Solution Methodology

We now discuss a solution strategy for our design problem when a cost-based distance function ($\delta^c_{\mathcal{M}^R_h}$) is used to determine the inexplicability of a plan. Given a plan π, such that, $\Gamma_{\mathcal{M}^R}(\mathcal{I}^R, \pi) \models \mathcal{G}^R$, the distance from an expected plan π' in the human model is given as $\delta^c_{\mathcal{M}^R_h}(\pi, \pi') =$

$$\begin{cases} \exp(|c^R_h(\pi) - c^R_h(\pi')|), & if\ \Gamma_{\mathcal{M}^R_h}(\mathcal{I}^R_h, \pi) \models \mathcal{G}^R_h \\ \infty, & otherwise \end{cases} \tag{5.8}$$

Here, we will use the set of plans that are optimal in the human's mental model as the expected plan set. This means that for calculating Equation 5.1, we do not require an additional minimization over the space of expected plans as every plan in the robot's model should be equidistant from every optimal plan in the human's

mental model (and the distance is infinity if the current robot plan is not executable in the human's mental model). For brevity, we refer to any plan with infinite inexplicable score as being invalid for a problem in $\mathbf{P}_{Exp}$. Also, we assume that the actions in both the models have unit costs. That is, $c_h^R(\pi) = c^R(\pi) = |\pi|$.

Proposition 2. $\forall i \in 1, \ldots, k, \pi, \pi' \in \Pi^*_{\mathcal{IE}(\cdot, \mathcal{M}_h^R(i), \delta_{\mathcal{M}_h^R(i)})}$,
$c^R(\pi) = c^R(\pi')$.

The above proposition states that all plans in $\Pi^*_{\mathcal{IE}(\cdot, \mathcal{M}_h^R(i), \delta_{\mathcal{M}_h^R(i)})}$ have equal costs in $\mathcal{M}^R(i)$ due to the assumption of unit costs. Therefore, while calculating the value for the objective function of $\mathcal{DP}_{Exp,\mathcal{D},\mathcal{T}}$, we can choose an arbitrary plan from $\Pi^*_{\mathcal{IE}(\cdot, \mathcal{M}_h^R(i), \delta_{\mathcal{M}_h^R(i)})}$ to calculate the term corresponding to the robot's cost.

5.4.1 Search for Optimal Design

To find the optimal solution for $\mathcal{DP}_{Exp,\mathcal{D},\mathcal{T}}$, we will perform a breadth-first search over the space of environment configurations that are achievable from the initial configuration through the application of the given set of modifications [51]. The performance of the search depends on the number of designs available. By choosing appropriate design strategies, significant scale up can be attained. Each search node is a valid environment configuration and the possible actions are the applicable designs. For simplicity, we convert the multi-objective optimization in Equation 5.2 into a single objective as a linear combination of each term associated with a coefficients α, β, and κ, respectively. The value of each node is decided by the aforementioned objective function. For each node, it is straightforward to calculate the design modification cost. However, in order to calculate the minimum inexplicability score and the robot's plan cost, we have to generate a plan that minimizes the inexplicability score for each explicable planning problem in that environment configuration. To achieve this, we

compile the problem of generating the explicable plan to a classical planning problem. We will discuss this compilation in the following subsection. Essentially, our search has two loops: the outer loop which explores all valid environment configurations, and the inner loop which performs search in a valid environment configuration to find a plan that minimizes the inexplicability score. At the end of the search, the node with best value is chosen, and the corresponding set of design modifications, ξ^*, is output.

One way to optimize our search over the space of environment configurations is to only consider the designs that are relevant to the actions in the optimal robot plans ($\Pi^*_{\mathbf{M}^R}$) and those in the human's expected plans ($\Pi^*_{\mathbf{M}^R_h}$) given the set of tasks. This can be implemented as a pruning strategy that prunes out designs that are not relevant to the actions.

5.4.2 Compilation for Most Explicable Plan

We show that generating the most explicable plan for a $\mathcal{P}_{Exp} = \langle \mathcal{M}^R, \mathcal{M}^R_h, \delta_{\mathcal{M}^R_h} \rangle$ is the same as generating an optimal plan, π^*_{mod}, for a transformed planning problem $\mathcal{P}_{mod}$. To this end, we leverage the compilation used by [90] and present a simplified version.

Definition 20. *Given an explicable planning problem, $\mathcal{P}_{Exp} = \langle \mathcal{M}^R, \mathcal{M}^R_h, \delta_{\mathcal{M}^R_h} \rangle$, the transformed planning problem is $\mathcal{P}_{mod} = \langle \mathcal{F}_{mod}, \mathcal{A}_{mod}, \mathcal{I}_{mod}, \mathcal{G}_{mod}, c_{mod} \rangle$, where,*

- $\mathcal{F}_{mod} = \mathcal{F}^R \cup \mathcal{F}^R_h$
- *For each $a_{mod} \in \mathcal{A}_{mod}$, $a_{mod} = \langle pre(a_{mod}), add(a_{mod}), del(a_{mod}) \rangle$, where:*
  $pre(a_{mod}) = \{f^R | f \in pre(a^R)\} \cup \{f^R_h | f \in pre(a^R_h)\}$
  $add(a_{mod}) = \{f^R | f \in add(a^R)\} \cup \{f^R_h | f \in add(a^R_h)\}$
  $del(a_{mod}) = \{f^R | f \in del(a^R)\} \cup \{f^R_h | f \in del(a^R_h)\}$

- $\mathcal{I}_{mod} = \{f^R | f \in \mathcal{I}^R\} \cup \{f_h^R | f \in \mathcal{I}_h^R\}$, *and*
  $\mathcal{G}_{mod} = \{f^R | f \in \mathcal{G}^R\} \cup \{f_h^R | f \in \mathcal{G}_h^R\}$

- *For each* $a_{mod} \in \mathcal{A}_{mod}$, $c_{mod}(a_{mod}) = c_h^R(a_h^R) = 1$

We label the fluents with different subscripts to denote that we maintain two separate copies of fluents in the transformed planning problem: i.e., for every $f \in \mathcal{F}$, there is robot's fluent, $f^R \in \mathcal{F}^R$ and the human's belief about it, $f_h^R \in \mathcal{F}_h^R$. We assume there is a one to one mapping between the actions in the robot's model and those in the human's mental model, so there are two versions of each action. The action transformation ensures that an action is executable by the robot if and only if its preconditions are satisfied in both $\mathcal{M}^R$ and $\mathcal{M}_h^R$, and it produces effects consistent with both models.

Proposition 3. *The problem* $\mathcal{P}_{mod}$ *produces a plan that solves* $\mathcal{P}_{Exp}$*, so that the following properties hold:*

- ***Soundness*** *A plan* π_{mod} *that solves* $\mathcal{P}_{mod}$ *is a valid solution for* $\mathcal{P}_{Exp}$*.*

- ***Completeness*** *For every valid plan that solves* $\mathcal{P}_{Exp}$*, there is a corresponding valid plan that solves* $\mathcal{P}_{mod}$*.*

- ***Optimality*** *A plan* π^*_{mod} *that solves* $\mathcal{P}_{mod}$ *optimally is the most explicable plan for* $\mathcal{P}_{Exp}$*.*

Proof. The transformed planning problem has the union of the constraints imposed by both $\mathcal{M}^R$ and $\mathcal{M}_h^R$. Given a plan π, such that, $\Gamma_{\mathcal{P}_{mod}}(\mathcal{I}_{mod}, \pi) \models \mathcal{G}_{mod}$, by the definition of the compilation, we also have $\Gamma_{\mathcal{M}^R}(\mathcal{I}^R, \pi) \models \mathcal{G}^R$ and $\Gamma_{\mathcal{M}_h^R}(\mathcal{I}_h^R, \pi) \models \mathcal{G}_h^R$. Hence, a plan π_{mod} that solves $\mathcal{P}_{mod}$ is a valid plan for $\mathcal{P}_{Exp}$.

From the definition of the inexplicability score for a plan π^R which is a valid solution to $\mathcal{P}_{Exp}$, we know that $\Gamma_{\mathcal{M}_h^R}(\mathcal{I}_h^R, \pi^R) \models \mathcal{G}_h^R$. Such a plan π^R solves both $\mathcal{M}^R$ and $\mathcal{M}_h^R$. Hence, π^R will satisfy, $\Gamma_{\mathcal{P}_{mod}}(\mathcal{I}_{mod}, \pi^R) \models \mathcal{G}_{mod}$. Therefore, for every valid plan that solves $\mathcal{P}_{Exp}$, there exists a corresponding plan that solves $\mathcal{P}_{mod}$.

Given $\mathcal{P}_{Exp}$, let π' be the most explicable robot plan (the plan with lowest inexplicability score) which is not an optimal plan for $\mathcal{P}_{mod}$. By definition of explicability, this means π' must be a valid plan for both $\mathcal{M}^R$ and $\mathcal{M}_h^R$. Further, by the completeness property, we know that π' must be a valid plan for $\mathcal{P}_{mod}$. This means that for a plan π^*_{mod} optimal in $\mathcal{P}_{mod}$, we have $c_h^R(\pi^*_{mod}) < c_h^R(\pi')$ (since $\mathcal{P}_{mod}$ uses c_h^R). Hence, $|c_h^R(\pi^*_{mod}) - c_h^{R*}| < |c_h^R(\pi') - c_h^{R*}|$, where c_h^{R*} is the cost of an optimal plan in $\mathcal{M}_h^R$ (and we know $c_h^{R*} \leq c_h^R(\pi^*_{mod})$ and $c_h^{R*} \leq c_h^R(\pi')$). This means $\mathcal{IE}(\pi^*_{mod}, \mathcal{M}_h^R, \delta_{\mathcal{M}_h^R}) < \mathcal{IE}(\pi', \mathcal{M}_h^R, \delta_{\mathcal{M}_h^R})$. This contradicts the initial assertion, proving that there is a one to one correspondence between optimal plans for $\mathcal{P}_{mod}$ and $\Pi^*_{\mathcal{IE}(\cdot, \mathcal{M}_h^R, \delta_{\mathcal{M}_h^R})}$. □

5.5 Evaluation

We will now demonstrate how the explicability value and design cost of the optimal solution evolve when optimizing for a single problem, multiple problems and multiple problems with a time horizon using the running example. We will also evaluate the performance of our approach on 3 IPC (International Planning Competition) domains and discuss the interplay between explicability and plan cost.

5.5.1 Demonstration

We use our running example from Figure 5.1a to demonstrate how the design problem evolves. We constructed a domain where the robot had 3 actions: *pick-up* and *put-down* to serve the items on a tray and *move* to navigate between the kitchen

and the booths. Some grid cells are blocked due to the tables and the robot cannot pass through these: cell(0, 1) and cell(1, 1). Therefore, the following passages are blocked: cell(0, 0)-cell(0, 1), cell(0, 1)-cell(0, 2), cell(0, 1)-cell(1, 1), cell(1, 0)-cell(1, 1), cell(1, 1)-cell(1, 2), cell(1, 1)-cell(2, 1). We considered 6 designs, each consisting of putting a barrier at one of the 6 passages to indicate the inaccessibility to the human (i.e. the design space has 2^6 possibilities).

For the following parameters: $\alpha = 1$, $\beta = 30$, $\kappa = 0.25$ and $\gamma = 0.9$, we ran our algorithm for three settings: (a) single explicable problem for $\mathcal{T} = 1$, (b) multiple explicable problems for $\mathcal{T} = 1$, and (c) multiple explicable problems for $\mathcal{T} = 10$. As mentioned before, (a) involved serving a new customer at a booth (say G_1) only once, (b) involved serving a new customer only once at either of the booths with equal probability and (c) involved serving each customer at most 10 times at either of the booths with equal probability. We found that for settings (a) and (b) no design was chosen. This is because these settings are over a single time step and the cost of installing design modifications in the environment is higher than the amount of inexplicability caused by the robot ($\beta > \alpha$). On the other hand, for setting (c), the algorithm generated the design in Figure 5.1b, which makes the robot's roundabout path completely explicable to the customers.

5.5.2 Domain setup

We used three IPC domains for evaluation: `Blocksworld`, `IPC-Grid` and `Driverlog`. For each domain, we created two versions: the robot's domain and the human's domain. We generated 20 design problems for each domain, and each had 3 planning problems with uniform probability distribution. All the experiments were run on an Ubuntu workstation with 64G RAM. We used Fast Downward with A* search and the *lmcut* heuristic [40] to solve the compiled planning problems. The variable

parameters in our implementation are α, β, κ (coefficients associated with the terms in the objective function), γ (discount factor) and $\mathcal{T}$ (time horizon). For all the domains we used actions and design modifications of unit cost.

For `Blocksworld`, the robot's domain was the original IPC domain, and the human's domain assumed that the robot can pick up multiple blocks simultaneously. The set of allowed designs ensured that stacking for every block was preceded by picking the block up from the table. This would reduce the inexplicability for the human as the only block that would be stacked is the one that was picked up from the table before stacking. In practice, this may involve notifying the human about the new rule. For `IPC-Grid`, the robot's domain was the original IPC domain and the human's domain assumed that diagonal movements were possible in the grid. We allowed design modifications that pruned diagonal actions. In actuality, this may involve notifying the human that diagonal actions are not possible at certain locations. For `Driverlog`, the robot's domain was the original IPC domain and the human's domain assumed that packages can be loaded and unloaded from the truck regardless of the location of the driver. We allowed modifications that required load and unload actions to occur only after a disembark action. This may again involve notifying the human about the new rules concerning load/unload actions.

5.5.3 Performance on IPC domains

For this objective, we set α, β and κ to 1.0, 0.25, 0.25 respectively for all domains i.e., we gave more weight to minimizing inexplicability. We set $\mathcal{T}$ to 1 and 10 and γ to 0.9. We allowed the search to run for 30 minutes per problem. If it ended within 30 minutes we output the optimal design modification, else we output the design modification which gave the best optimization value (or total cost) among the explored nodes. Note that in the `IPC grid`, we restricted the set of applicable designs

Domain	Horizon	Metrics	Design Size	Inexplicability			Plan Cost			Total Cost			Time Taken (secs)
				w/o Design	w Design	% Difference	w/o Design	w Design	% Difference	w/o Design	w Design	% Difference	
Blocksworld	1	Avg	1.25	14.11	2.18	-84.54	8.69	9.52	9.58	16.28	4.87	-70.07	1800
		SD	0.79	16.86	0.92	-	1.39	1.85	-	17.11	1.38	-	
	10	Avg	1.25	91.90	14.20	-84.54	8.69	9.52	9.57	113.63	38.33	-66.27	
		SD	0.78	109.80	5.98	-	1.39	1.85	-	112.36	9.59	-	
IPC-Grid	1	Avg	0.75	3571.84	1455.39	-59.25	24.84	24.84	0	23326.29	1461.79	-93.73	1800
		SD	0.44	12043.62	4428.98	-	3.01	3.01	-	78444.61	4429.19	-	
	10	Avg	0.75	23264.19	9479.32	-59.25	24.84	24.84	0	23326.29	9541.61	-59.09	
		SD	0.44	78442.72	28846.93	-	3.01	3.01	-	78444.61	28848.86	-	
Driverlog	1	Avg	0.8	2.26	1.6	- 29.14	8.46	9.17	8.46	4.37	4.09	- 6.39	219.42
		SD	0.77	0.54	0.57	-	0.59	0.89	-	0.61	0.54	-	
	10	Avg	1.2	14.70	8.93	-39.28	8.45	9.71	14.76	35.85	33.50	- 6.57	
		SD	0.69	3.54	2.78	-	0.59	0.97	-	4.30	3.94	-	

Table 5.1: We Report the Impact of Design Modifications on Inexplicability Score, Plan Cost and Total Cost. We Also Report the Average and Standard Deviation Values for the Three Optimization Terms in the Objective Function along with the Run Time.

at each node to the ones that affect the current expected human plan. To show the impact of design modifications, we computted the inexplicability score, the plan cost, and the total cost for the most explicable plan in the initial model without any design modification. To compare the impact of longitudinality, we compute these for single step horizon and multi-step horizon.

In Table 5.1, we report the results for the 3 domains. By comparing the inexplicability score with and without design, we see that the inexplicability always decreases as expected. For Blocksworld and IPC-Grid, the percentage decrease is the same for one-step and multi-step horizon; this is because the same set of designs were the best solutions found for both settings (under the time-limit) and the values got multiplied

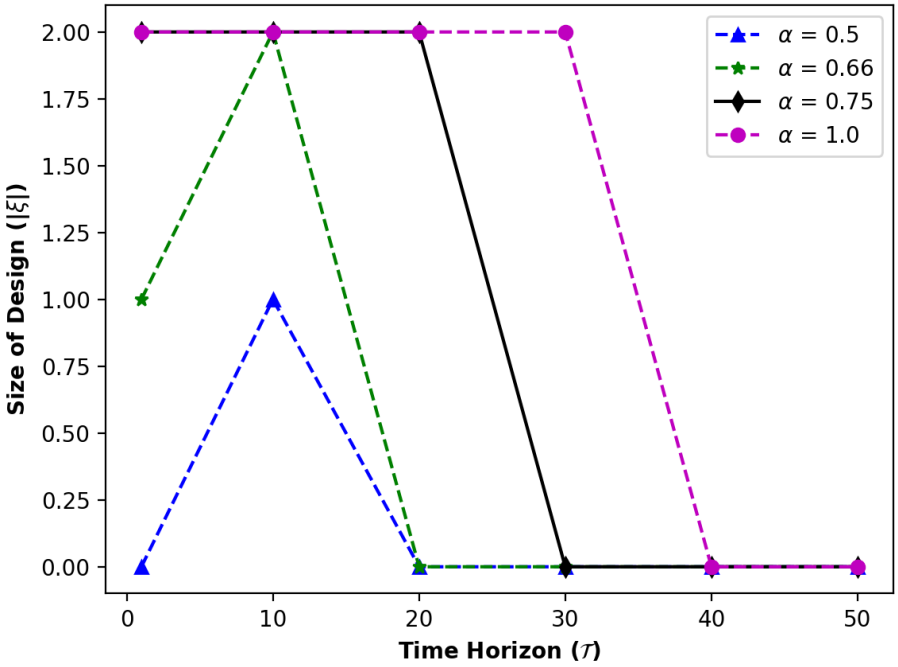


Figure 5.3: The Plot Shows the Impact of Inexplicability Score Coefficient (α) on Design Size in the Solutions over Different Time Horizons for a `Driverlog` Problem.

with the value of $\mathcal{T}$. On the other hand, for `Driverlog`, there were different designs selected, as is evident from the values. By comparing the plan cost with and without design, we can see that for `Blocksworld` and `Driverlog`, there is a substantial increase in the plan cost. This is because for these two domains, the designs ensured an action could be performed only after execution of another action. In this case, the robot bears additional cost for improving the explicability. On the other hand for `IPC-Grid`, the action pruning strategy removed actions from the human's mental model and therefore there is no increase in the plan cost. Similarly, by comparing the total cost with and without design, we can see that there is a significant decrease in the total cost after applying design modifications. This is because the optimization chooses design modifications that minimize the overall cost associated with the initial model.

5.5.4 Interplay Between Inexplicability Score and Plan Cost

To study the interplay between inexplicability score and plan cost, we experimented with a $\mathcal{DP}_{Exp,\mathcal{D},\mathcal{T}}$ problem in the `Driverlog` domain. We used discount factor $\gamma = 0.9$ and design cost coefficient $\beta = 0.25$. We tested the impact of different inexplicability score coefficient values (α: 0.5, 0.66, 0.75, 1) on the number of design choices in an optimal solution given different time horizons $\mathcal{T}$: 1, 10, 20, 30, 40, 50. At most two design choices were allowed in the solution.

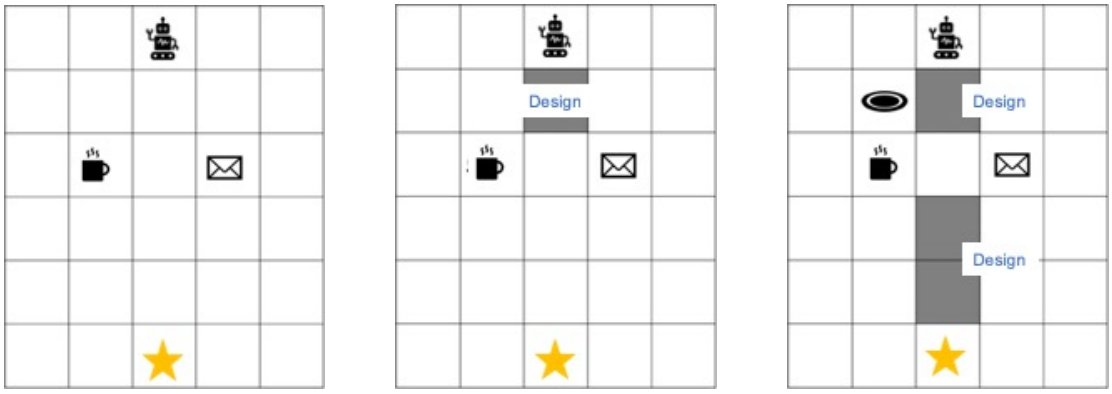
In Figure 5.3, we report the impact on the size of design modifications. Recall that, the discount factor γ denotes the probability that the human will not ignore the inexplicability of the behavior. Therefore, when γ is set to 0.9, the optimization prioritizes reduction in inexplicability score. Given that the design cost coefficient $\beta = 0.25$ is low, even with single time step horizon $\mathcal{T} = 1$, designs are found that improve the explicability of the robot's behavior as shown in Figure 5.3. However, the designs in the `Driverlog` domain lead to an increase in the cost of the robot plan (due to additional disembark actions). Given a long time horizon ($\mathcal{T} = 50$), the cost overhead borne by the robot for being explicable becomes greater than the impact of the inexplicability score on the human. Hence no designs are found at $\mathcal{T} = 50$ for any of the α values. If explicability of the robot's behavior is desired for longer horizons, this can be achieved by setting α to a high value. This shows the inherent interplay between the inexplicability of the behavior and the additional plan cost borne by the robot to reduce inexplicability.

Environment Design for Legibility and Predictability

This section details a general formulation for the environment redesign problem to facilitate legible and predictable robot behaviors. We will first discuss the notion

of environment design problem for communicative behaviors in general. And then we will outline the legibility and predictability scores that can be plugged into the general problem of environment design for communicative behaviors. We discussed the notion of offline goal and plan legibility in Chapter 4. However, in this section, our discussion on legibility will focus on online settings – i.e. legibility score with respect to partially executed plan prefixes of the robot. Further, this legibility score will be more general i.e. defined with respect to candidate robot models (as against only goals or only plans). In this section, we will also discuss the notion of predictability with respect to plan prefixes. Predictable robot behavior allows the human observer to easily anticipate robot's possible plan completion to an executed plan prefix. We briefly touched upon this notion in Section 4.4.1, while discussing the connection between plan legibility and predictability. As we have seen in this chapter, one of the biggest advantages of environment design is that the process of generation of a desired behavior is offloaded from the robot onto the design process, and therefore is a useful tool for facilitating desirable robot behaviors in structured settings. We will now see the problem of environment design for legibility and predictability with an illustrative example,.

Illustrative Example Consider an office setting where an office assistant robot, responsible for delivering items such as coffee or mail to the employees, is about to be deployed (Figure 5.4a). The robot (*actor*) will be supervised by office security guards (*observer*) who have worked with previous generation office assistant robots and have some expectations regarding their functions. In particular, they expect the robot to carry one item at a time (i.e. either mail or coffee) and each robot generally has a strong preference on the order in which it picks up these items (though the order changes from robot to robot). Unknown to the guards, the new model adds



(a) (b) (c)

Figure 5.4: The Office Assistant Domain: (a) the Original Domain; (b) to Induce Legible Behavior, We Can Add Dividing Walls to Constrain the Agent and Help the Observer Reduce Uncertainty in Their Mental Model; And (c) to Induce Predictable Behavior We Can Reduce Uncertainty about the Item Being Picked up by Including a Tray That Allows the Agent to Pick Up Both of the Objects.

more flexibility to the robot by (1) removing the need for the robots to have fixed preference on the order to pick up items and (2) installs a coffee cup holder that allows the robot to carry both mail and coffee at the same time. Now if we allow the new robot to simply act optimally in the original setting, it might unnecessarily confuse the observers.

If the robot was built to generate communicative behavior, it will change its behavior (and possibly settle for suboptimal decisions in its own model) in order to address these model differences. However, the same effect can be achieved if the designers who are *deploying* the robot also designed the environment to ensure that decisions of the new robot remain interpretable to the occupants of the office.

If the designers wish to prioritize legibility, the aim is to help the user differentiate between the models as early as possible, one way to do it would be to disable the coffee

holder and then introduce obstacles as shown in Figure 5.4b. As for predictability, the objective is to allow the user to be able to predict the entire plan as early as possible. One possible design for this scenario is to disable the coffee holder and provide the robot with a tray that allows the robot to carry both of the items at the same time. The observer can see the tray and realize that the robot can place both items in the tray. Further, we need to add additional obstacles to restrict the space of possible plans that can be done by the robot in a cost-optimal manner (Figure 5.4c).

Problem Setting

In this setting, the robot operates in an environment while being observed by the human. A communicative planning problem is a tuple, $\mathcal{P}_{Comm} = \langle \mathcal{M}^R, \mathbf{M}_h^R, Comm \rangle$, where, $\mathcal{M}^R$ is the robot's model of the task, $\mathbf{M}_h^R$ is the observer's mental model of the robot, represented by a set of candidate task models that the observer believes that the robot has, and $Comm : \Pi_{\mathcal{M}^R} \rightarrow \mathbb{R}$ is the communicative behavior score that is used to evaluate robot's plans (where $\Pi_{\mathcal{M}^R}$ is the space of robot plans). Interestingly, we do not require that $\mathcal{M}^R \in \mathbf{M}_h^R$ – i.e. the models in $\mathbf{M}_h^R$ can be different from $\mathcal{M}^R$ in all possible aspects (e.g. state space, action space, initial state and goals). The solution to $\mathcal{P}_{Comm}$ is a plan or policy that not only solves $\mathcal{M}^R$ but also satisfies some desired properties of communicative behaviors (measured through the score). The score could reflect properties like legibility or predictability of the plan.

Legibility With legibility, the objective of the robot is to inform the observer about its model – i.e. reduce the size of $\mathbf{M}_h^R$. A robot's behavior is considered to be perfectly legible if it can be derived from only one model in $\mathbf{M}_h^R$. In an online setting, the longer it takes for a plan prefix to achieve this, the worse is the plan's legibility. This notion thus helps the observer narrow down their belief over the possible robot models as

quickly as possible.

Predictability The objective of the robot with predictability is to generate the most disambiguating behavior – i.e. given the robot's plan prefix, the observer should be able to predict its completion. These predictions would be in terms of cost-optimal completions of a given prefix in the possible problems in the mental model. This means that if there exists the same unique completion in all of the models then *the plan is predictable even though not legible.* The shorter the length of the disambiguating plan prefix, the better the predictability of the plan. An empty prefix would thus correspond to the most predictable plan.

Environment Design for Communicative Behaviors

We now present a general formulation for the design problem for communicative behaviors. Given an environment design, we assume that the robot is a rational agent and therefore is incentivized to generate cost-optimal plans. Let the set of cost optimal plans of the robot be $\Pi^*_{\mathcal{M}^R}$. A cost-optimal plan solution to $\mathcal{M}^R$ can exist anywhere on the spectrum of legibility and predictability from high to low. Therefore, we need a measure to quantify the communicative score for the robot's set of cost-optimal plans. To that end, we introduce the worst-case communicative score *wcc* as follows:

Definition 21. *The* ***worst-case communicative score*** $wcc(\cdot)$*, for* $\mathcal{P}_{Comm}$ *is defined as*

$$wcc(\mathcal{P}_{Comm}) = \min_{\pi \in \Pi^*_{\mathcal{M}^R}} \quad Comm(\pi) \tag{5.9}$$

$Comm(\cdot)$ is instantiated for each type of communicative behavior separately and is discussed in detail at the end of this section. The higher the score, the better

the communicative ability of the behavior (in terms of either of two properties). Therefore, the worst-case communicative score is the minimum communicative score of a cost-optimal plan of the robot.

We can now define the design problem for communicative behavior. When a modification is applied to the environment, both the robot's model and the observer's mental model are modified, thereby changing the worst-case communicative score of the robot's cost-optimal plans for the given model. Let $\mathcal{P}$ denote the set of valid configurations in the real environment. Although $\mathcal{M}^R \in \mathcal{P}$, problems in $\mathbf{M}_h^R$ might not necessarily be in $\mathcal{P}$ if the observer has incorrect or infeasible notions about the robot's model. Therefore, we represent the set of configurations that the observer thinks are possible as $\widetilde{\mathcal{P}}$, and $\mathbf{M}_h^R \subseteq \widetilde{\mathcal{P}}$.

Definition 22. *The **design problem for communicative behavior**, DP-Comm, is a tuple* $\langle \mathcal{P}^0_{Comm}, \Delta, \Lambda^R, \Lambda^R_h \rangle$ *where,*

- $\mathcal{P}^0_{Comm} = \langle \mathcal{M}^{R^0}, \mathbf{M}_h^{R^0}, Comm \rangle$ *where* $\mathcal{M}^{R^0} \in \mathcal{P}$ *and* $\mathbf{M}_h^{R^0} \subseteq \widetilde{\mathcal{P}}$ *are the initial models.*
- Δ *is the set of modifications that can be applied to the environment.* ξ *is a sequence of modifications.*
- $\Lambda^R : \Delta \times \mathcal{P} \to \mathcal{P}$ *and* $\Lambda^R_h : \Delta \times \widetilde{\mathcal{P}} \to \widetilde{\mathcal{P}}$ *are the model transition function that specify the resulting model after applying a modification to the existing models.*

The set of possible modifications includes modifying the set of states, action preconditions, action effects, action costs, initial state and goal. Each modification $\xi \in \Delta$ is associated with a cost, such that, $C(\xi) = \sum_{\xi_i \in \xi} C(\xi)$. After applying ξ to both $\mathcal{M}^{R^0}$ and $\mathbf{M}_h^{R^0}$, the resulting robot decision making problem model and observer mental model are represented as $\mathcal{M}^{R^{|\xi|}}$ and $\mathbf{M}_h^{R^{|\xi|}}$ respectively.

Let $\mathcal{P}_{Comm}^{|\xi|}$ be the modified communicative planning problem after applying the modification ξ to $\mathcal{P}_{Comm}$. Our objective here is to solve *DP-Comm* such that the worst-case communicative score of $\mathcal{P}_{Comm}$ is maximized. Apart from that, the design cost of ξ has to be minimized, as well as the cost of a plan π^R that solves $\mathcal{M}^{R|\xi|}$.

Definition 23. *A **solution to** **DP-Comm**, is a sequence of modifications ξ with*

$$\min(-wcc(\mathcal{P}_{Comm}^{|\xi|}), C(\xi), cost(\pi^R)) \tag{5.10}$$

This completes the general framework of design for communicative behaviors. In the following, we will look at specific instances of design for the different notions of communicative behaviors.

Design for Legibility

In order to be legible, the robot's plan has to reveal its task to the observer as early on as possible. Therefore, the legibility of a plan is inversely proportional to the length of its shortest prefix that has unique cost optimal completion for more than one problem in the observer's mental model.

Definition 24. *The **legibility score** $Leg(\cdot)$ of a robot's plan, π^R, that solves $\mathcal{M}^R$ is defined as follows:*

$$Leg(\pi^R) = \min_{\tilde{\pi}^R \in \tilde{\Pi}_{\mathcal{M}^R}} e^{-|\tilde{\pi}^R|} \tag{5.11}$$

such that $\exists(\mathcal{M}_h^{R^i}, \mathcal{M}_h^{R^j}) \in \mathbf{M}_h^R, i \neq j$ with unique cost optimal completion of $\tilde{\pi}^R$ in each model, and $\tilde{\Pi}_{\mathcal{M}^R}$ is the set of all prefixes of π^R. Plugging this scoring function in Equation 21 allows us to instantiate the design problem for legibility.

Goal Recognition Design The work on goal recognition design (GRD) [47] is a special case of the design problem for legibility. The GRD problem involves a robot

and an observer where the observer's mental model consists of planning models that have the exact same state space, actions and initial state as the robot's planning model. However, each planning model in the observer's mental model has a different goal. The robot's true goal is one of them, and the objective of GRD problem is to redesign the environment, such that, the true goal of the robot is revealed to the observer as early as possible. The communicative planning problem defined here is a general one, where the observer's mental model can be different in all possible ways from the robot's actual planning model.

Design for Predictability

In order to be predictable, the plan has to be the most-disambiguating plan among the set of plans the observer is considering – i.e. the observer should be able to predict the rest of the plan after seeing the prefix. Therefore, predictability of a plan is inversely proportional to the length of its shortest prefix which ensures only one optimal completion solving only a single problem in the observer's mental model. We can quantify the predictability score as follows:

Definition 25. *The* ***predictability score*** $Pred(\cdot)$ *of a robot's plan* π^R *that solves* $\mathcal{M}^R$ *is defined as follows:*

$$Pred(\pi^R) = \min_{\tilde{\pi}^R \in \tilde{\Pi}_{\mathcal{M}^R}} e^{-|\tilde{\pi}^R|} \tag{5.12}$$

such that $\exists! \pi \; \exists \mathcal{M}_h^{R^i} \in \mathbf{M}_h^R$ where π is an optimal completion of $\tilde{\pi}^R$, and $\tilde{\Pi}_{\mathcal{M}^R}$ is the set of all prefixes of a plan π^R. Plugging this scoring function in Equation 21 allows us to instantiate the design problem for predictability.

Connection to Plan Recognition Design The predictability problem corresponds to the plan recognition design (PRD) problem [72]. However, our proposed

framework in terms of possible observer models subsumes the plan library based approaches in being able to support a generative model of observer expectations.

5.6 Concluding Remarks

In this work, we bridge the gap between past works on environment design and those on generation of explicable behavior. We present a novel framework of environment design for explicability. As we saw, the notion of environment design is more suitable option to facilitate explicability, when there is repeated execution of tasks or when there are multiple tasks in the environment. This allows us to explore a novel trade-off that arises between the one-time cost of design and the repeated cost overhead incurred by the robot for generating explicable behavior. In general, the design modifications can also be software changes that only affect the robot's capabilities. In prior works on explicable plan generation, the underlying setting considered a one-time interaction between a human and a robot. In this work, we relaxed this assumption and explored the notion of inexplicability given repeated interactions. Further, we also saw the general problem of environment design for communicative behaviors like legibility and predictability. And that the problems of goal recognition design and plan recognition design fall out as special cases of design for legibility and design for predictability frameworks respectively.

Chapter 6

PLANNING FOR OBFUSCATORY BEHAVIOR

In this chapter, we will focus the discussion on obfuscatory strategies that a robot can employ in an adversarial environment. So far, we have seen how a robot can synthesize interpretable behaviors while it is interacting with a cooperative human observer. However, in the real world not all of the robot's interactions may be of purely cooperative nature. The robot may come across entities of adversarial nature while it is completing its tasks in the environment. In such cases, the robot may have certain secondary objectives like privacy preservation, minimization of information leakage, etc. in addition to its primary objective of achieving the task. Further, in adversarial settings, it is not only essential for the robot to minimize information leakage but also to ensure that this process of minimizing information leakage is secure. Since, an adversarial observer may use diagnosis to infer sensitive information from the gleaned observations and then use that information to interfere with the robot's objectives.

To prevent leakage of sensitive information, the robot should be capable of generating behaviors that can obfuscate the sensitive information about its goals or plans. A robot can hide its true goal from an adversarial observer by making several goals seem plausible at the same time. Thus the true objective stays obfuscated from the observer. Additionally, if the obfuscation is not secure, the adversary can try to glean sensitive information from multiple different executions of the robot, therefore the obfuscation has to also be secure. The robot can choose to obfuscate its actions as well apart from its goals. We show that these obfuscatory behaviors can be captured using the same controlled observability planning framework (COPP) introduced in Chapter 4. Additionally, we will also look at another approach for goal obfuscation

which balances the obfuscation with amount of resources available. Here the robot's objective is to obfuscate the true goal for as long as the resource budget allows.

6.1 Related Work

There are several prior works which discuss the problem of privacy preservation in distributed multi-agent systems [8; 65; 7]. A recent work on privacy for multi-agents by Maliah *et al.* [68] is complementary to our approach, as they consider problems where the model needs to be protected from the team members but goals and the behavior are coordinated. In contrast, we consider problems where the models are public but goals and behavior need to be protected.

The problem of goal obfuscation is directly related to plan and goal recognition literature [78; 79; 31; 88; 47; 49; 77]. Traditional plan recognition systems have focused on techniques where actions being executed can be observed directly. Although since we model these obfuscatory behaviors in the COPP framework, the observational equivalence resulting from the many-to-one formulation of the observation function $\mathcal{O}$ introduces, in effect, noisy action-state observations. This, in turn, complicates plan recognition. More crucially, the robot uses the observational equivalence to actively hinder the ease of plan recognition.

A few recent works have explored the idea of obfuscation in adversarial settings from the goal recognition aspect [50; 70]. Keren *et al.* [50] propose a solution that obfuscates a goal by choosing one of the candidate goals that have the maximum non-distinct observation sequence in common with the true goal. Our approach generalizes this notion with respect to k number of candidate goals, and additionally our approach also guarantees secure goal obfuscation for these k candidate goals, such that, even if the adversary tries to run the algorithm with different inputs, there is no additional information leakage about the robot's true goal.

The concept of deception has also been explored in the context of adversarial environments. While the concept of obfuscation deals with the notion of confusing the adversary with several decoy candidates, the concept of deception [70] involves making one of the decoy candidates more likely than the robot's true objective/activities. Thus, with deceptive strategies, in order to deceive an adversarial observer, it is crucial to have access to their goal or plan recognition models. By incorporating the predictions of the goal or plan recognition model, the robot can synthesize behavior that deceives the adversarial observer into believing that the decoy candidate is the true candidate. In the prior literature, synthesis of deceptive behaviors with respect to robot's goals has been studied in path planning scenarios where the adversary has full observability of robot's activities [70]. In order to successfully deceive an adversarial observer, the robot's plan has to end when a decoy goal is achieved. However, in reality, the robot has a primary objective of achieving its true goal. Therefore, in cases where the observer has full observability of the robot's activities, fully deceptive behavior may be hard to synthesize.

6.2 Goal Obfuscation

The problem of goal obfuscation arises when the robot wants to obfuscate its true goal from the adversary. This is possible in settings where the adversarial entity is unaware of the robot's true goal but is aware of the possible candidate goals that the robot may try to achieve in the given domain. This problem setting is also part of the controlled observability planning problem, introduced in Chapter 4. Therefore, this is also an offline setting, where the adversary only gets the observation sequence once the robot has executed its actions. In fact, the goal obfuscation setting is similar to the setting seen in the goal legibility problem. However, here because of the existence of an adversarial observer, instead of being legible with respect to its true

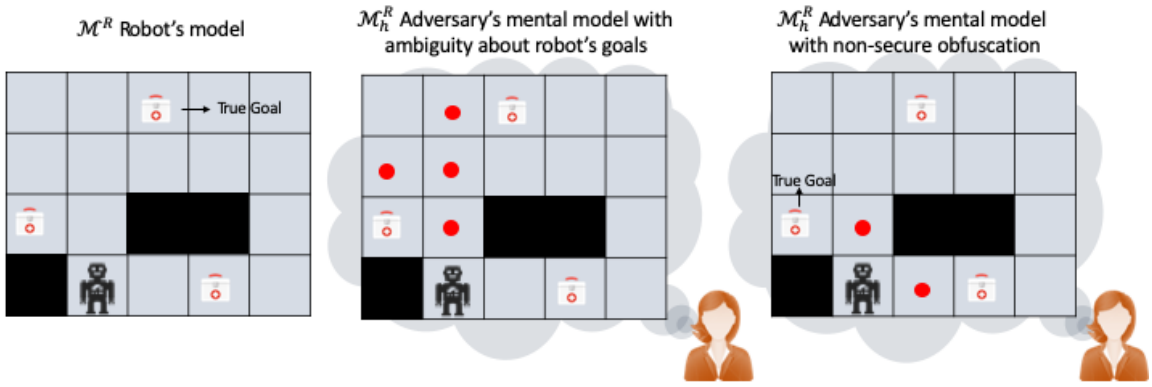


Figure 6.1: Illustration of Impact of Goal Obfuscation and Secure Goal Obfuscation on Human's Mental Model.

goal, the robot obfuscates its true goal. In order to obfuscate effectively, the robot needs to have access to the adversarial observer's sensor model. Such a sensor model defines the type of observations the adversary may get. The adversary may use these observations resulting from the robot's behavior to infer some sensitive information about the robot's goals. Depending on the granularity of the observations available to the adversary, the problem of goal obfuscation becomes easy or complex. That is with fine-grained observations, goal obfuscation might be harder to achieve or may even be infeasible in some cases, whereas with coarse-grained observations, it might be easier for the robot to obfuscate its true goal. Further, the adversary may be able to use the information gleaned from observations to interfere with or hamper the robot's activities. Therefore, in this setting, the robot is tasked with the additional objective of preventing the adversary from learning sensitive information about its goal.

Definition 26. *A **goal obfuscation planning problem**, is a controlled observability planning problem, i.e., a $\mathcal{P}_{\mathcal{CO}}$, where, $\mathcal{G} = \{G_A \cup G_1 \cup \ldots \cup G_{n-1}\}$, is the set of n goals where G_A is the true goal of the robot, and $G_1, \ldots, G_{n-1}$ are decoy goals.*

6.2.1 Computing Goal Obfuscatory Plans

A solution to a goal obfuscation planning problem is a *k-ambiguous* plan. Recall that, since the observer receives observations of the robot's behavior, each plan is associated with a corresponding observations sequence. Therefore, the objective of a *k-ambiguous* plan is to make the observation sequence consistent with at least k goals, out of which $k-1$ are decoy goals, such that, $k \leq n$. This is done by taking actions towards robot's true goal, such that, the corresponding observation sequence exploits the observer's belief space in order to be consistent with multiple goals. That is, a *k-ambiguous* plan produces a final belief state consistent with its observation sequence where at least k goals are possible. The $k-1$ decoy goals can be chosen specifically to maximize the obfuscation. Figure 6.1 illustrates goal obfuscation for $k = 2$ in the middle image. Here the true goal of the robot to pickup the medkit is marked in the leftmost image. The adversarial observer has partial observability of the direction in which the robot moves (say, the observer receives coarse observations from the GPS sensor, which does not reveal the exact direction of the movement). In order to obfuscate its true goal from the adversarial observer, the robot performs a 4-step k-ambiguous plan (for k=2), as shown in the middle image.

To define a *k-ambiguous* plan, we use the notation for belief state, introduced in Chapter 4, under Section 4.2.

Definition 27. *A plan, π_k, is a* ***k-ambiguous plan****, if $\Gamma(\mathcal{I}, \pi_k) \models G_A$ and the final belief state, $b_n \in BS(\pi_k, \mathcal{I})$, satisfies the following, $|G \in \mathcal{G} : \exists s \in b_n, s \models G| \geqslant k$, where $1 \leqslant k \leqslant n$.*

Definition 28. *An observation sequence $O_k = \langle o_1, \ldots, o_n \rangle$ is* ***k-ambiguous observation sequence*** *if it is an observation sequence emitted by a k-ambiguous plan.*

The underlying algorithm for generating a *k-ambiguous* plan is the same as Algo-

rithm 3 introduced in Chapter 4. However, the goal test and heuristic function are customized to ensure that the goal obfuscation constraints are met.

Goal test

The goal test ensures that the solution plan does not end unless all k goals occur in the observer's belief state. Specifically, it validates whether the true goal has been achieved in the robot's true state and also whether the $k-1$ goals have been achieved in the belief state. If a plan that can achieve the true goal with the given k goals doesn't exist, we restart the search with next combination of k goals (inclusive of true goal) among the total n candidate goals. In the worst case, the value of k can be reduced until a solution is found.

Heuristic function

Here the heuristic strategy is to generate a plan where the observer's final belief state satisfies k goals, making it a *k-ambiguous* plan. We use two heuristic functions to estimate a node's cost: $h(s) = h_{G_A}(s) + h_{\mathcal{G}_{k-1}}(b)$, where the first part computes the heuristic cost to the true goal from the robot's actual state, and the second part computes the heuristic cost to $k-1$ goals from the belief induced by the emitted observation. This is computed by finding the *max* of heuristic cost to each of the $k-1$ goals. The heuristic value of a state is computed as the total of these two values. Thus the heuristic helps to guide the search towards at least k goals in the observer's belief.

6.3 Secure Goal Obfuscation

The above approach to synthesizing goal obfuscating behavior is not secure. In the sense that it reveals information when the following assumptions are not satisfied

[64; 97]: (1) adversary knows the actor's algorithm: this means that the adversary has access to the algorithm and knows how the algorithm works. For example, the adversary knows what states are generated by the algorithm. (2) Independence of inputs: this means that the adversary can run the algorithm with different inputs. For example, the adversary can set different goals as true goal to check the variability of the output. Figure 6.1, rightmost subfigure illustrates how an adversary can rerun the algorithm with different input and glean information about the true goal. By setting the true goal to some other candidate goal, the adversary can run the algorithm and find out the robot's original goal which gave the observation sequence.

6.3.1 Computing Secure Goal Obfuscatory Plans

In order to synthesize a secure goal obfuscating behavior, we can introduce some randomization in our solution synthesis process, that would prevent leakage of additional information. To that end, we randomly choose a set of k goals (inclusive of true goal) from n candidate goals. Then randomly pick a goal from this set as a "true goal" and run the algorithm. Let's call this goal as the decoy true goal. We have to maintain the plans to each of the k goals that are consistent with the observation sequence. Then by performing a post processing step, we can retrieve the plan that reaches the robot's true goal and is consistent with the observation sequence produced by the plan to the decoy true goal. This process ensures that even if the adversary runs the algorithm again with either the same or different input (different goal as true goal, or different combination of k goals), the identity of the true goal remains *k-ambiguous*.

In this setting, a perfectly rational adversary would be one that has unbounded computational resources, and can use them to make perfect inferences. In other words, that is the worst case scenario for our algorithm. The following proposition shows

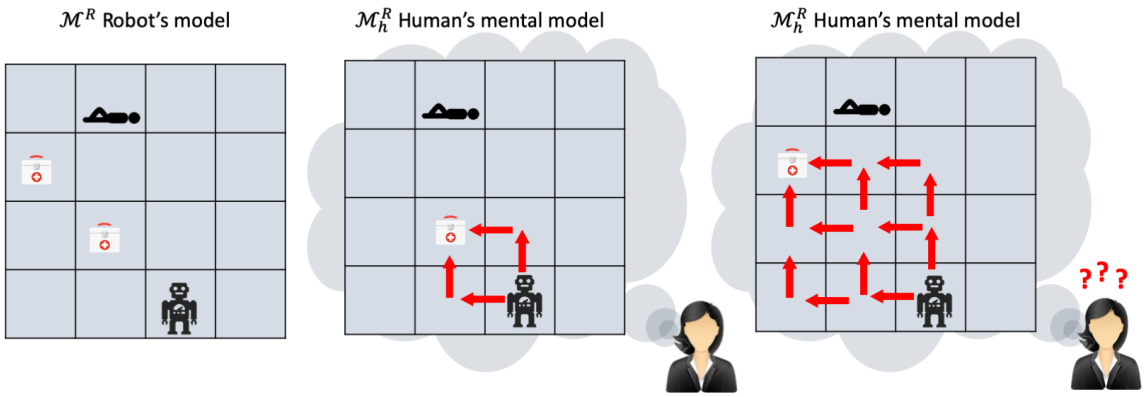


Figure 6.2: Illustration of the Impact of Plan Obfuscation on Human's Mental Model.

that our algorithm provides *k-ambiguous* guarantee even in the worst case, such that a perfectly rational adversary remains uncertain about the true goal.

Proposition 4. *Let $\mathcal{H}$ be a perfectly rational adversary. Given a goal obfuscation planning problem for an robot $\mathcal{R}$. Plans computed using Algorithm 3 with k-ambiguous goal test and heuristic function are such that, $\mathcal{H}$ will be able to infer the true goal with probability $\leqslant 1/k$.*

Let's say goal G_i from the set $\mathcal{G}_k$ is chosen randomly to be the decoy true goal. The observation sequence O_i is obtained by running the algorithm with $G_A = G_i$. The adversary can run the algorithm with each goal from $\mathcal{G}_k$ to search for G_i. But G_i can only be the robot's true goal with probability $1/k$. Therefore the adversary can detect the true goal with probability $\leqslant 1/k$.

Note that the goals can be specified using sub-formulae. This makes the input specification much smaller depending on how many states are in the domain that are consistent with the sub-formulae. In the least constrained domain, that may be exponential in the number of propositions not used.

In this problem setting, the adversarial observer is aware of the robot's goal since there is a single candidate goal. The robot, however, obfuscates its activities towards its goal thereby hiding sensitive information about its activities. We refer to this behavior as plan obfuscatory behavior. The robot can achieve a plan obfuscatory behavior by computing a plan whose observation sequence conforms to a set of plans that are diverse in terms of their actions, thereby making it hard for the adversarial observer to guess the actions executed by the robot. For instance, in Figure 6.2, the goal of the robot is to pick up a medkit and treat the victim on the disaster site. It has two choices of medkits to pick from. If it chooses the medkit as shown in the rightmost sub-figure, it can maximize the diversity with respect to its actions. Here, the multiple different paths to the medkit produce the same observation sequence (due to the underlying coarse GPS sensor model, which doesn't reveal the direction of the robot's movement), thus hiding the robot's true actions from the adversarial observer.

Definition 29. *A **plan obfuscation planning problem** is a tuple,* $\mathcal{P}_{PO} = \langle \mathcal{D}, \mathcal{G}_{PO}, \Omega, \mathcal{O} \rangle$*, where,* $|\mathcal{G}_{PO}| = 1$.

6.4.1 Computing Plan Obfuscating Plans

The solution to a plan obfuscation problem is an ℓ-diverse plan. An ℓ-diverse plan is a plan whose observation sequence is consistent with at least ℓ diverse plans to the goal, such that, the diverse plans are at least d distance away from each other. In order to compute an ℓ-*diverse* plan, we need to keep track of the plans that are consistent with the observation sequence and reach the goal. To compute the diversity between all the pairs of plans consistent with the observation sequence, a plan distance measure

[93; 74] like action distance, causal link distance, state sequence distance (introduced in Chapter 3) can be used. Our approach can use any valid plan distance. We now define an ℓ-*diverse* plan using the notion of d-distant pairs of plans, where d is the distance between the two plans, which was introduced in Chapter 4. We will also use the notion of belief plan set (BPS) introduced in Chapter 4, which is a set of plans that are consistent with the belief sequence of a given plan.

Definition 30. *A BPS induced by plan p starting at s_0 is* ***minimally d-distant****, $d_{min}(BPS(p, s_0))$, if $d = \min_{p1,p2 \in BPS(p,s_0)} \delta(p1, p2)$.*

Definition 31. *A plan, π_l, is an* ***ℓ-diverse plan****, if for a given value of d and distance function δ, $d_{min}(BPS(\pi_l, \mathcal{I})) \geq d$, $|BPS(\pi_l, \mathcal{I})| \geq \ell$, where $\ell \geq 2$ and every plan in $BPS(\pi_l, \mathcal{I})$ achieves the goal in $\mathcal{G}$.*

In order to generate a solution plan to the plan obfuscation problem, we use Algorithm 3 as before. Again, the goal test and heuristic function are customized to ensure that there are at least ℓ diverse plans to the true goal that are consistent with the observation sequence and the minimum distance between these plans is at least d.

Goal test

To ensure the plans in BPS, induced by an ℓ-*diverse* plan, can achieve the goal in $\mathcal{G}$, we check whether at least ℓ plans are reaching the goal or not and whether the minimum distance between plans in BPS is at least d. Also in order to ensure termination of the algorithm, there is a cost-bound given as input to the algorithm.

Heuristic function

Apart from minimizing the heuristic cost to the goal, the customized heuristic given below also maximizes the d of $d_{min}(BPS(p, s_0))$ induced by plan p starting at s_0. This increases the minimum distance between the plan pairs. This distance is computed using a plan distance measure.

$$h(s) = h_{G_A}(s) - d_{min}(BPS(p, s_0)) \tag{6.1}$$

6.5 Empirical Evaluation of COPP Problem Variants

We now present an empirical evaluation of two COPP problem variants discussed in this chapter. Through empirical evaluation, we intend to investigate the following objectives **(1)** comparison between run time and plan costs across the two problem variants versus the optimal solution to the true goal, **(2)** cost overhead incurred by the decoy true goal randomization step in goal obfuscation problem, **(3)** impact of Δ in Algorithm 3 on goal obfuscation. The domain and experimental setup for these experiments is same as that in Section 4.5.

6.5.1 Results

The evaluation results are presented in Tables 6.1 and 6.2 and Figure 6.3. We modified the STRIPS planner Pyperplan [1] to implement our algorithms. We used the `hsa` [53] heuristic of Pyperplan because it gave the best results in terms of computation time. We ran the experiments with $k = 3$, $\ell = 3$, $d_{min} = 0.25$ for all the domains. We used action distance measure to compute the distance between plans. We ran our experiments on 12 core Intel Xeon CPU with an E5-2643 v3@3.40GHz processor with a 64G RAM with 20 minutes time-out. The following number of problems reached time-out before a solution could be found for goal legibility and obfuscation variants together: 8/50 in the `Blocksworld`, 11/50 in `Logistics` and 6/50

Domain	Metrics	*k-amb secure*	*ℓ-div*	opt
Blocksworld	Time	196.9	140.88	0.99
	Length	12.33	10.84	9.6
Logistics	Time	206.13	100.04	8.31
	Length	28.63	25.92	23.74
Driverlog	Time	145.03	111.99	1.54
	Length	14.33	13.59	11.16

Table 6.1: Empirical Evaluation for Goal Obfuscation and Plan Obfuscation Solved Using the Optimization Presented in Algorithm 3 Versus the Optimal Plan Solution (Opt Column) to the True Goal. We Report the Average Time (in Seconds) and the Average Plan Length.

in `Driverlog`. These problems were excluded from the results to ensure consistency across problems.

The Table 6.1 presents a comparison between different variants of our framework. For *k-amb secure*, we randomize the decoy true goal and use BPS to retrieve the plan to the true goal. The process of maintaining BPS for every node is expensive and leads to higher run time, as can be seen in the Table. Another reason for higher run time is that the planner is working with multiple goals, as $k = 3$ in our experiments. Our approach is secure against replay attacks (attacks where the adversary tries running algorithm with different inputs). For *ℓ-div*, the run time cost comes from maintaining BPS for each node. We also compare the time taken to compute an optimal plan to the true goal represented by the *opt* column, which is comparatively

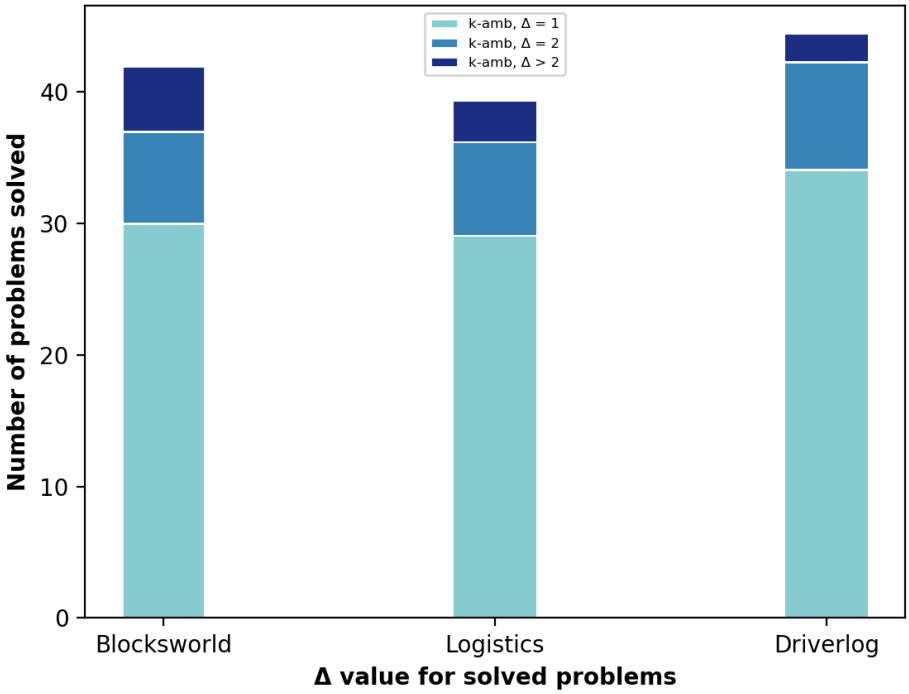


Figure 6.3: Empirical Evaluation of Δ in Algorithm 3 for Goal Obfuscation and Goal Legibility Variants. We Report the Number of Problem Instances Solved for Different Values of Δ.

Domain	*k-amb* w/o	*k-amb* w/	*k-amb* secure
Blocksworld	122.7	140.62	196.9
Logistics	159.28	181.74	206.13
Driverlog	112.93	123.02	145.03

Table 6.2: Empirical Evaluation to Report the Average Time (in Seconds) for Different Versions of *k-ambiguous* Algorithms. *k-amb w/* and *w/o* Do Not Use *BPS* and Report Time with and Without the Additional Post-processing Step. *k-amb Secure* Uses *BPS* to Provide Robust Solutions to Replay Attack.

faster since the planner does not take into account the information being inferred by the adversary.

The Table 6.2 shows performance comparison across different techniques of achieving *k-ambiguous* solutions. For *k-amb w/*, we do not maintain BPS for every node, instead we retrieve the plan to the true goal by filtering the belief space using the observation sequence. This can sometimes lead to incorrect solutions (because path consistency is not checked) but it can be computed faster than the solution that maintains BPS (*k-amb robust*). For *k-amb w/o*, we show the run times without the extra filtering step to retrieve the plan to the true goal, i.e., the run time for generation of k-ambiguous observation sequence. From Figure 6.3, we see that for goal obfuscation, most of the solutions are obtained for $\Delta = 1$ for all the 3 domains. Thus showing that the solutions for goal obfuscation can be obtained for smaller value of Δ parameter of Algorithm 3.

Secure Non-Biased Goal Obfuscation

In this goal obfuscation framework, the robot has access to a limited resource budget. The robot's objective is to securely obfuscate the true goal for as long as the resource budget allows and in a cost-effective manner. The general problem of goal obfuscation involves hiding the true goal of the agent from the observer by making the observation sequence "consistent" with k candidate goals. In this section, we introduce a stronger definition of consistency, such that, even when the solution computation mechanism is run with either of the k goals as a "true goal", we get the same observation sequence. In the secure goal obfuscation approach involving randomization, different candidate goals in the set of k goals will give different observation sequences, but due to the randomization itself and the post-processing step, the true goal continues to remain obfuscated. Here in contrast, all the k candidate

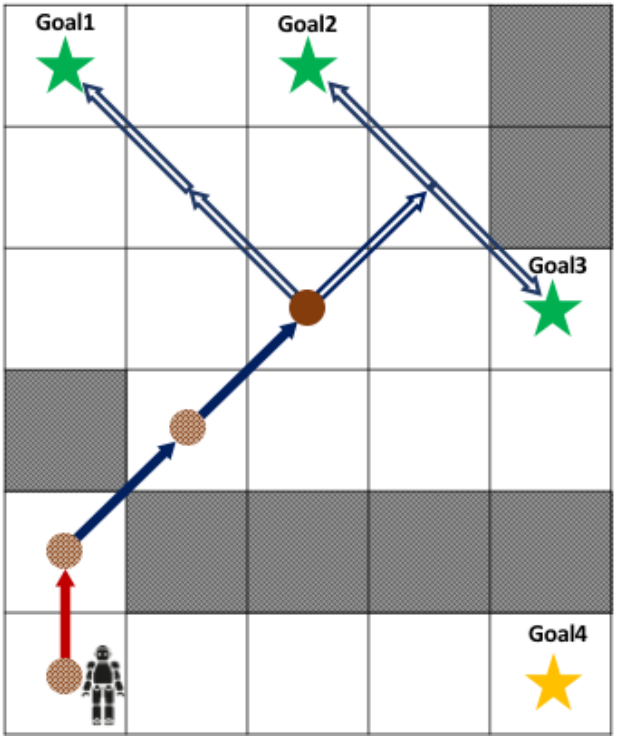


Figure 6.4: A `Gridworld` Example Illustrating Our Definition of Privacy. With an Observation Model That Distinguishes Diagonal and Orthogonal Actions, the Observer Sees the Same Sequence of Observations for All the Three Goals Regardless of the Agent's True Goal.

goals give the same observation sequence. Thus, the solution observation sequence makes all the k goals *equally likely* for as long as possible. Even this goal obfuscation approach is secure to replay attacks. Further, this approach can compute the cheapest solution that satisfies the secure obfuscation guarantee. In the following paragraph, we illustrate the intuition of this approach using the following example.

Consider the `gridworld` in Figure 6.4. Assume each cell allows movement in all 8 adjacent cells with equal cost. The initial state is at cell (0, 0). The candidate goals are given by Goal1, Goal2, Goal3 and Goal4, and Goal2 is the robot's true goal. With $k = 3$, we select Goal1, Goal2 and Goal3 because they have greater similarity in

terms of landmarks. The observation model emits two types of observation symbols for each cell: o_1 (marked by blue arrow) if the next cell is diagonally adjacent and o_2 (marked by red arrow) otherwise. In the Figure 6.4, we can see three plans where each reaches a candidate goal. Each of these plans produce the same observation sequence given by $\{o_2, o_1, o_1, o_1, o_1\}$. Each of these plans is a secure goal-obfuscated plan, where $k = 3$. The observation sequence is the same regardless of the true goal. A secure non-biased goal-obfuscated plan can be achieved by finding the cheapest path to a closest point that is equidistant (in terms of remaining steps) from each of the goals. In the example, the equidistant state at cell (3, 4), is 2 steps away from each goal. From this equidistant state, we check if there exists an observation sequence which is common to the 3 goals and reaches the goals in only 2 steps. From cell (3, 4), the observation sequence $\langle o_1, o_1 \rangle$ satisfies the requirement. The four equidistant states for the three candidate goals are marked in brown color.

If the observation model allows for full observability, such that, each grounded action is mapped to a unique observation symbol, then a fully obfuscated plan is not possible. In such cases, our aim is to obfuscate the true goal for as long as possible by computing a plan that achieves an equidistant state, such that the cost from the equidistant state to each of the goals is minimized. We also discuss a method for choosing k goals among a set of n candidate goals, such that, we can minimize the solution cost and maximize the overall obfuscation. In the following sections, we discuss this approach in detail. We also establish upper and lower bounds on the cost of solutions which ensure secure non-biased goal obfuscation. We present evaluation of our approaches using three International Planning Competition (IPC) domains, namely, `Blocksworld`, `Logistics` and `Zenotravel`.

Problem Setting

We assume all the actions in the planning problem are of uniform cost and equal to 1. In other words, the length of the plan is equal to the cost of the plan and an optimal plan is a shortest length plan for the given planning problem. We adapt the goal obfuscation problem definition to also include robot's resource budget as follows:

Definition 32. *A **resource bounded goal obfuscation** is a goal obfuscation planning problem, $\mathcal{P}_{\mathcal{GO}} = \langle \mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G}, \Omega, \mathcal{O}, \mathcal{R} \rangle$, where, $\mathcal{R}$ is the cost budget of the robot.*

Properties of Secure Non-biased Goal-Obfuscation A secure non-biased goal-obfuscated plan is a solution to $\mathcal{P}_{\mathcal{GO}}$, that minimizes the bias of the observation sequence towards the true goal. With respect to a chosen set of k goals, such that, $\mathcal{G}_k \subseteq \mathcal{G} \wedge G_A \in \mathcal{G}_k$, if we can make the observation sequence associated with the robot's plan identical for all the k goals, then there is no bias and the true goal is securely obfuscated. If this is not possible, we minimize the bias by maximizing the length of identical observations corresponding to plans for each goal in k. The value of k can be $k \leq n$. We will now distill three important properties of a secure non-biased goal-obfuscated plan: **(1)** For each goal in $\mathcal{G}_k$, there should exist a plan which achieves the goal in l number of steps, such that, l is within the resource budget, **(2)** The observation sequence associated with k plans that each achieve a goal in $\mathcal{G}_k$ should be the same, **(3)** The observation sequence should not be biased by any goal in $\mathcal{G}_k$. The first and second properties are straight-forward. The third property states that the robot should be able to compute the same observation sequence while generating a plan solution to any of the k goals. If the above mentioned properties are followed, the observation sequence associated with the plan does not disclose any specific information about the true goal. We will refer to a plan that satisfies the above properties as a secure non-biased goal obfuscatory plan.

Computation of Secure Non-Biased Goal-Obfuscatory Plans

In this section, we will discuss our approach for computing a secure solution to a resource bounded goal obfuscation planning problem. Given the properties a secure non-biased goal obfuscatory plan, we can say the following:

Definition 33. *A **secure non-biased goal obfuscatory planner**, computes a plan solution $\pi_{\mathcal{P}_{\mathcal{GO}}}$, and an observation sequence $O_{\mathcal{P}_{\mathcal{GO}}}$, to a $\mathcal{P}_{\mathcal{GO}} = \langle \mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G}_k, \Omega, \mathcal{O}, \mathcal{R} \rangle$, where, $\Gamma(\mathcal{I}, \pi_{\mathcal{P}_{\mathcal{GO}}}) \models G_A$, such that, $\forall G' \in \mathcal{G}_k$, if G' is set as the true goal of the robot, there exists a plan $\pi'_{\mathcal{P}_{\mathcal{GO}}}$, and an observation sequence $O'_{\mathcal{P}_{\mathcal{GO}}}$, where $\Gamma(\mathcal{I}, \pi_{\mathcal{P}_{\mathcal{GO}}}) \models G' \wedge O'_{\mathcal{P}_{\mathcal{GO}}} = O_{\mathcal{P}_{\mathcal{GO}}}$.*

Therefore, given a set of k goals, a secure non-biased goal-obfuscatory planner will generate the same observation sequence when any of the k goals is set as a true goal. For a given problem, if there does not exist a secure non-biased goal obfuscated plan (say the observation model doesn't allow for it), then the planner computes a partial plan starting from initial state that satisfies the the secure non-biased obfuscation guarantee for as long as possible. This partial plan guarantees secure obfuscation for the entirety of its length (although not until the goal). In such cases, we compute partial secure plans which maximize the coverage of obfuscation from the initial state. The process of computing a secure non-biased goal-obfuscated plan consists of two phases: **(1)** selection of k goals from the set of n candidate goals and **(2)** computation of a secure non-biased goal-obfuscated plan and observation sequence. The first phase selects k candidate goals (inclusive of true goal) that have higher similarity with each other. And then, the second phase computes a plan whose observation sequence is consistent with all of the goals present in the chosen set of k goals.

Decoy Goal Selection We choose a set of $k-1$ candidate goals such that these goals have higher similarity with the true goal and with each other. We use a landmark based measure to compute the similarity between the candidate goals. There is a prior work that has made use of disjunctive landmarks for diverse planning [9]. In automated planning, for a given problem instance each landmark is a subset of fluent instantiations that every plan must satisfy in order to solve the problem. The intuition behind using landmarks is that, the goals with common landmarks will go through similar states/actions and thereby introduce inherent ambiguity in the plans that achieve the candidate goals. We can choose $\mathcal{G}_k$ as follows:

1. For each goal, $G \in \mathcal{G}$, extract the set of landmarks and add each distinct landmark to a set $\mathcal{L}$.

2. Initialize each landmark $L \in \mathcal{L}$ with an empty list.

3. For each distinct landmark L, if it belongs to a goal G, add G to L's list.

4. Order the landmarks in $\mathcal{L}$ in decreasing order of the number of associated goals.

5. Select all the landmarks in $\mathcal{L}$ that are associated with at least k goals. If no such landmarks exist then starting from the first select all landmarks until there are k unique goals.

6. Order the unique goals in the decreasing order of their frequency in the selected landmarks.

7. If the true goal appears in first half then starting from the first goal (otherwise, starting from the last goal), divide the goals in groups of unique k goals.

8. Select the group of k goals that includes the true goal.

Given the set of k goals, we can obtain a lower bound on the cost of the secure non-biased goal-obfuscated plan.

Proposition 5. *For given $\mathcal{G}_k$, let π_{G_i} be the optimal plan to reach the farthest G_i in $\mathcal{G}_k$. If π_k is a secure non-biased goal obfuscated plan that solves $\mathcal{P}_{\mathcal{GO}}$ then,*

$$cost(\pi_k) \geqslant cost(\pi_{G_i})$$

The above proposition states that the cost of a solution to $\mathcal{P}_{\mathcal{GO}}$ cannot be cheaper than the optimal cost to reach the farthest goal in the set of k goals. This proposition can be useful in a resource sensitive setting, the decoy goals can be chosen such that the lower bound of the solution cost is smaller than the available cost-bound. Note that in our setting, all the actions have unit cost.

Secure Non-biased Goal-Obfuscated Plan Once we have chosen the set of k goals, the next step is to compute a secure non-biased goal-obfuscated plan. In order to compute it, we do the following: **(1)** we first compute a set of states that are equidistant to each of the goals in $\mathcal{G}_k$ and **(2)** then we compute a bounded length plan from the equidistant state to a goal, such that the observation sequence is same for plans reaching other goals from the equidistant state. We call it bounded length belief plan. As stated in the third property listed in Subsection 6.5.1, the observation sequence should not be biased by any particular goal in $\mathcal{G}_k$. In order to ensure this property, we use an equidistant state. An equidistant state has equal number of remaining observations to each of the goals in $\mathcal{G}_k$. This state removes bias of the search towards any particular goal. That is, regardless of the true goal, the search visits the same states. Therefore the adversary does not receive any new information even when the adversary knows how the algorithm works. Another advantage of computing equidistant state is that it reduces the length of bounded length belief plan,

Algorithm 5 Computation of Equidistant States

Input: $\mathcal{P}_{GO}$ **Output:** list of equidistant states

1: $open \leftarrow$ `Priority_Queue()`; $closed \leftarrow \{\}$; $equidistant \leftarrow$ `Priority_Queue()` ▷ Open, closed and equidistant lists
2: $h\_diff, h\_max \leftarrow$ `Heuristic_Computation(`$\mathcal{G}_k$`)`; $open.push(\mathcal{I}, h\_max)$
3: **if** $h\_diff = 0$ **then**
4: $equidistant.push(\mathcal{I}, h\_max)$
5: **end if**
6: **while** $open \neq \emptyset$ **do**
7: $s \leftarrow open.pop()$
8: $closed \leftarrow closed \cup s$
9: **for** $a, s' \in successors(s)$ **do**
10: $g(s') \leftarrow g(s) + cost(a)$
11: $h\_diff, h\_max \leftarrow$ `Heuristic_Computation(`$\mathcal{G}_k$`)`
12: **if** $s' \notin$ open **and** $s' \notin$ closed **then**
13: $open.push(s', g(s') + h\_max)$
14: **if** $h\_diff = 0$ **then**
15: $equidistant.push(\mathcal{I}, g(s') + h\_max)$
16: **end if**
17: **else**
18: $g(s') < g^{prev}(s')$
19: **if** $s' \notin$ open **then**
20: $closed \leftarrow closed \setminus s'$
21: $open.push(s', g(s') + h\_max))$
22: **if** $h\_diff = 0$ **then**
23: $equidistant.push(\mathcal{I}, g(s') + h\_max)$
24: **end if**
25: **else**
26: update priority to $g(s') + h\_max$
27: **end if**
28: **end if**
29: **end for**
30: **end while**
31: **return** $equidistant$
32: **procedure** `Heuristic_Computation(`$\mathcal{G}_k$`)`
33: **for** $G \in \mathcal{G}_k$ **do**
34: $h_{\mathcal{G}_k} \leftarrow h_{\mathcal{G}_k} \cup h_G$
35: **end for**
36: $h\_diff \leftarrow max(h_{\mathcal{G}_k}) - min(h_{\mathcal{G}_k})$; **return** $(h\_diff, max(h_{\mathcal{G}_k}))$

as the belief space search can be computationally expensive to the robot. Formally, an *equidistant state* and a *bounded length belief plan* are defined as follows:

Definition 34. *An **equidistant state**, e, is a state in the state space of a $\mathcal{P}_{G\mathcal{O}}$ from which there are d number of steps/observations remaining to each of the k goals.*

Each of the k goals can be achieved in equal number of steps from an equidistant state. When the observation model allows partial observability, the observer operates in the belief space. For every observation emitted, the observer's belief space is updated to reflect all the possible states consistent with the observation sequence. For example, in Figure 6.4, the observation $o1$, is consistent with the robot being in any of the 4 diagonally adjacent cells. On the other hand, when the observation model allows full observability, the belief space only consists of robot's true state. Note that, as mentioned before, in case of full observability only partial plan, that is plan up to the closest (to the goals) equidistant state is secure. In the case of partial observability, we compute bounded length belief plan.

Definition 35. *A **bounded length belief plan**, π_{G_A}, is a plan of length, d, associated with an observation sequence, O_k, where $\Gamma(e, \pi_{G_A}) \models G_A$, then $\forall\ G \in \mathcal{G}_k\ \exists\ \pi_G$ of length d and associated with observation sequence O_k, where $\Gamma(e, \pi_G) \models G$. Here d is the number of remaining steps from e to each of the goals.*

From an equidistant state, we perform a blind search for bounded length d in the belief space. After d steps if all the goals are found such that the observation sequence is same then we output the bounded length belief plan and its observation sequence.

A secure non-biased goal-obfuscated plan is then generated by computing a cheap plan to an equidistant state, e and then computing a bounded length belief plan from e to the goals. While computing the shortest path from initial state to the equidistant state, we also compute the belief associated with the equidistant state.

Algorithm 6 Computation of Bounded Length Belief Plan and Observation Sequence

Input: $\mathcal{P}_{GO}$, *equidistant* **Output:** obs-sequence

$h_{min} \leftarrow \infty$; $farthest_{equi} \leftarrow \emptyset$
while *equidistant* $\neq \emptyset$ **do**
  $s_{equi}, h_{equi} \leftarrow equidistant.pop()$
  if $h_{equi} < h_{min}$ **then**
    $farthest_{equi} \leftarrow s_{equi}$
  end if
  $b_{equi} \leftarrow$ belief with respect to initial state
  open $\leftarrow$ `Priority_Queue()`; *closed* $\leftarrow \{\}$ ▷ Open and closed list
  $open.push(\langle s_{equi}, b_{equi} \rangle)$; $closed \leftarrow closed \cup b_{equi}$
  while *open* $\neq \emptyset$ **do**
    $s, b \leftarrow open.pop()$
    if $g(s) > g(s_{equi}) + h_{equi}$ **then**
      break
    end if
    if $b \models$ GOAL-TEST$(\mathcal{G}_k)$ **then**
      return obs-sequence for s
    end if
    for $a, s' \in successors(s)$ **do**
      $o \leftarrow \mathcal{O}(a, s')$
      $b' \leftarrow$ Belief-Generation(b, a, o)
      $g(s') \leftarrow g(s) + cost(a)$
      if $b' \notin closed$ **then**
        $open.push(\langle s', b' \rangle)$
        $closed \leftarrow closed \cup b'$
      end if
    end for
  end while
end while
return obs-sequence for $farthest_{equi}$

Then the initial belief for the bounded length belief plan is populated with the belief associated with the equidistant state. The computation of the first part is done by performing a state space search. We search the state space to compute a list of equidistant states. If from a given state, the nearest and the farthest of the k goals have the same number of remaining steps then we add it to the list of equidistant states. The details of this procedure are given in Algorithm 5. The computation of second part constitutes performing a bounded length blind search in the belief space starting from each equidistant state until the solution is found. We process the list of equidistant states obtained in the first part and start our bounded length belief search from the closest equidistant state to the goals. The details of this procedure are given in Algorithm 6.

Given the set of equidistant states in the entire state space of $\mathcal{P}_{\mathcal{GO}}$, we can obtain a upper bound on the cost of the secure non-biased goal-obfuscated plan.

Proposition 6. *For given $\mathcal{G}_k$, let $\mathcal{E} = \{e_1, \ldots, e_n\}$ be the set of all the equidistant states for $\mathcal{P}_{\mathcal{GO}}$, such that, $\forall e_i \in \mathcal{E}, \pi^{\mathcal{I}}_{e_i}$ be the plan from initial state to e_i and $\pi^{e_i}_{G}$ be the plan from e_i to a $G \in \mathcal{G}_k$. If π_k is a secure non-biased goal obfuscated plan that solves $\mathcal{P}_{\mathcal{GO}}$ then,*

$$cost(\pi_k) \leqslant \underset{e \in \mathcal{E}}{\operatorname{argmax}} \; cost(\pi^{\mathcal{I}}_{e}) + cost(\pi^{e}_{G}) \tag{6.2}$$

The above proposition states that the cost of a solution to $\mathcal{P}_{\mathcal{GO}}$ is bounded by the cost of a plan via the costliest equidistant state. This proposition can be useful in a resource sensitive setting, the set of equidistant states can be filtered further before starting belief space search such that the upper bound on the solution cost is smaller than or equal to the available cost-bound.

Empirical Evaluation

We evaluate the scope and usability of this approach. Our empirical evaluation measures the following metrics:

1. The impact of different observation models on the extent of obfuscation.
2. The trade-off between additional cost and extent of obfuscation possible.
3. The comparison between run time and plan costs for goal obfuscation planning versus optimal planning.

In the following, we will discuss the domains used, and the setup of the problems and observation models for each of these. Then we will discuss the observations and results derived for each of the three evaluation objectives.

Domains and Experimental Setup We use three IPC domains, namely `Blocksworld, Logistics` and `Zenotravel` to evaluate our approach.

For each of the domains, we randomly generated 50 problem instances. For the `Blocksworld` domain, we generated problems with 4 to 8 blocks and towers of maximum height 4 for both initial and goal states. For the `Logistics` domain, we generated problems with goals consisting of 2 to 6 facts. For the `Zenotravel` domain, we generated problems with goals consisting of 2 to 6 facts. We generated 5 random candidate goals (n=5) for each problem.

In order to explore the first evaluation objective, we generated both partially and fully observable observation models for all the three domains. The partially observable models have many-to-one mapping of action-state pairs to observation symbols. For the sake of simplicity, we used lifted action names as observation symbols. Therefore, the grounded actions taken and associated states are mapped to the corresponding lifted action names. For the `Blocksworld` domain, the observation symbols

were *pickup, putdown, stack, unstack*. For the `Logistics` domain, the observation symbols were *load-truck, unload-truck, load-airplane, unload-airplane, drive-truck, fly-airplane*. Finally for the `Zenotravel` domain, the symbols were *board, debark, fly, zoom, refuel*. The fully observable models have one-to-one mapping, that is the observer is aware of the actions taken and the states reached by the agent.

In order to balance the run-time between the state space search and the belief space search, we go back and forth between these two searches. For a given problem, when the heuristic value reaches some threshold (say, reduces by 50% of its value from the initial state), we start processing the states in the equidistant queue. We then run the belief space search for the states in equidistant queue until they are exhausted. Once the equidistant queue has been exhausted, we restart the state space search again to find further equidistant states exhaustively. After which, we again start with belief space search. We can modulate the heuristic threshold according to the problem types and desired objectives. This strategy can be especially helpful when the state space is large, if the planner spends all its time resource in exploring the state space, there will be no resource left for exploring the belief space. For the experiments, we kept the threshold of 50%.

Results The evalution results are presented in Tables 6.3, 6.4 and 6.5. We modified the STRIPS planner pyperplan [1] to implement our algorithms. To compute equidistant states we used `lmcut` heuristic [1] . We ran the experiments with $k = 3$ for all the domains. We ran our experiments on 2.7 GHz Intel Core i7 processor with 16 GB RAM. The experiments were run with 15 minutes time-out for each problem instance. The following number of problems reached time-out before a solution could be found: 4 out of 50 problems in the `Blocksworld` domain, 19 out of 50 problems

[1]By using `lmcut` heuristic, we approximate the equidistant states since it is not the same as the actual optimal cost $h^*(\cdot)$.

Domain	Metrics	Observation Models	
		Many-to-One	One-to-One
Blocksworld	Avg obfuscated length (%)	81.50	24.31
	Avg run time	46.85	58.39
Logistics	Avg obfuscated length (%)	84.06	62.93
	Avg run time	244.68	256.18
Zenotravel	Avg obfuscated length (%)	73.75	48.65
	Avg run time	216.14	224.14

Table 6.3: Empirical Evaluation for Different Types of Observation Models. We Report Average Percentage of Obfuscated Plan Length, and the Average and Standard Deviation of Time Taken (in Seconds) to Compute the Obfuscated Plan.

Domain	Metrics	Cost-bounds	
		+20%	+50%
Blocksworld	Avg obfuscated length (%)	71.90	77.75
Logistics	Avg obfuscated length (%)	73.25	83.58
Zenotravel	Avg obfuscated length (%)	65.81	69.87

Table 6.4: Empirical Evaluation to Explore the Cost Versus Obfuscation Trade-off. We Examine the Extent of Obfuscation for Different Cost-bounds. We Report Average Percentage of Obfuscated Plan Length.

in `Logistics` and 22 out of 50 problems in `Zenotravel`. These problems were not considered in the results.

Domain	Metrics	Optimal Plan	Obfuscated Plan
Blocksworld	Avg plan cost	5.64	6.57
	Avg run time	0.11	46.85
Logistics	Avg plan cost	26.40	27.94
	Avg run time	14.87	244.68
Zenotravel	Avg plan cost	8.3	10.79
	Avg run time	10.18	216.14

Table 6.5: Empirical Evaluation to Explore Differences in Optimal Plan to Goal and Obfuscated Plan to Goal. We Report Average Plan Cost and Average, Standard Deviation of Time Taken in Seconds to Compute the Solution to the Goal.

Table 6.3 Through Table 6.3, we examine how the different observation models affect the length of obfuscated part of the plan solution. It is not always possible to find a completely secure non-biased goal-obfuscated plan. In such cases, we output a partial solution. Here the obfuscated solutions were found without any resource constraints. For the fully observable observation models, we see that the `Blocksworld` domain had the least obfuscated solutions. For the partially observable observation models, we see that `Logistics` domain had the most obfuscated solutions.

Table 6.4 Through Table 6.4, we evaluate the impact of constraining the additional cost on the length of obfuscated parts of the solution. We ran experiments with cost bound of 1.2 and 1.5 times the optimal cost of the true goal. By only incurring 20% additional cost, we are able to generate plans with substantial portions obfuscated. Increasing the additional cost does not dramatically improve the obfuscation performance. This highlights the importance of the observation model and candidate goal

selection in the resource bounded goal obfuscation problem.

Table 6.5 Through Table 6.5, we compare the run-time and cost differences between optimal and goal-obfuscated plans. For all the three domains, the plan cost of obfuscated plans is slightly higher than optimal plans. Although the main difference lies in the amount of time taken to compute the obfuscated plans. Given there is enough time available to compute obfuscated plans, the plan cost incurred for the obfuscation guarantee is not too high.

Note that, as the number of goals in $\mathcal{G}_k$ is reduced, it becomes easier to find a set of equidistant states. Also, note that, we set a timeout of 15 minutes for each problem. As the the size of $\mathcal{G}_k$ is reduced and the time-out limit is increased more problems can be solved. The agent can use the upper and lower cost-bounds (given by Propositions 1 and 2) to decide the resource and goal obfuscation trade-off, and compare this trade-off over a variety of sizes of $\mathcal{G}_k$ to choose an appropriate size k.

6.6 Concluding Remarks

In this chapter, we discussed how the obfuscatory behaviors can also be framed in the controlled observability planning framework, which was introduced in Chapter 4. We discussed the goal obfuscation and plan obfuscation behaviors that allow the robot to obfuscate its true goal and its activities respectively from an adversarial observer. We also discussed how the goal obfuscation algorithm can be made secure to replay attacks, which allow an adversary with access to the algorithm to query it with different inputs. Further, we also presented the problem of resource bounded goal obfuscation. Depending on the resource budget, the robot can modulate the amount of obfuscation it needs. It can then synthesize a secure non-biased goal-obfuscatory plan which allows the robot to choose a set of decoy goals such that,

the bias of the observation sequence towards either of the goals is reduced. Using these decoy goals, a secure non-biased goal-obfuscatory plan is found. We performed empirical evaluation to analyze the effectiveness of both the COPP variants as well as the resource bounded goal obfuscation problem.

Chapter 7

PLANNING FOR SIMULTANEOUSLY OBFUSCATORY AND LEGIBLE BEHAVIOR

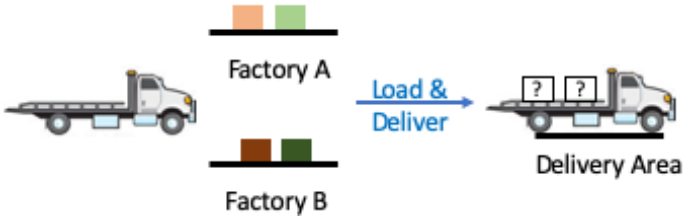
There are several environments which involve multiple types of observers, where some are of adversarial nature while some others are of cooperative nature. In such cases, the robot has to ensure that its behavior is simultaneously legible to cooperative observers and obfuscatory to adversarial ones. These observers in the environment may have access to asymmetric information about the robot's objectives. Typically, different observers have disparate sensing capabilities owing to differing levels of prior knowledge and communication. For example, a robot may establish semaphore actions that are more meaningful to its allies than to its adversaries. In this work, we develop methods that allow a robot to synthesize behavior that is simultaneously obfuscating towards known adversaries and legible towards known allies. We assume that different types of observers have differing sensor models (for instance, as a result of prior communication).

Several prior works have looked at generating legible behaviors for cooperative robots and obfuscating behaviors for adversarial robots [28; 50; 102; 70; 66; 86; 59]. By taking the observer's sensing capabilities into account, legible behavior helps the robot to convey necessary information to a cooperative observer, whereas, obfuscating behavior helps in hiding sensitive information from an adversarial observer. These approaches, however, assume that the observers in the environment are either entirely cooperative or entirely adversarial. In real-world settings of strategic importance, a robot might encounter both types of observers simultaneously, which would necessitate synthesizing a behavior that is simultaneously legible to friendly entities and

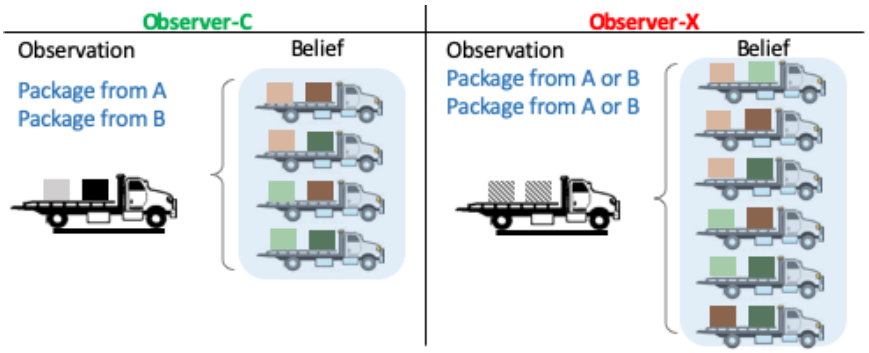
obfuscatory to adversarial ones. For instance, in soccer, a player may perform feinting trick to confuse an opponent while signaling a teammate. Similarly, in military planning, troops may execute maneuvers that signal their intended course of action to their allies while confusing the enemy. Synthesizing a single behavior that is legible and obfuscatory to different agents, presents significant technical challenges. In particular, the agent may have to be deliberately less legible to friends so that it can be effectively more obfuscatory to the adversaries. This problem gives rise to a novel optimization space that involves trading-off the amount of obfuscation achieved for adversaries with the amount of legibility desired for friends.

We now present a novel problem framework called mixed-observer controlled observability planning problem, MO-COPP, that allows an autonomous agent to simultaneously control information yielded to both cooperative and adversarial observers while achieving its goal. Essentially, this framework models and exploits situations where different observers have differing sensing capabilities, which result in different "perceptions" of the same information. We then present two solution approaches to solve this problem. In the first solution approach, we propose a novel integer programming (IP) encoding, which provides an optimal solution given a fixed time horizon. This involves maximizing the amount of obfuscation while maximizing the amount of legibility with respect to the agent's objectives. In the second approach, we present a heuristic-guided search algorithm, which leverages an approach [59] that assumes entirely cooperative or entirely adversarial settings. Through theoretical and empirical analysis, we explore the properties of the two solution approaches. Additionally, we show that for mixed-observer settings our solution approaches add significant value over approaches that consider either cooperative observers or adversarial observers in isolation.

(a) The actor's goal is to deliver two packages to the delivery area:



(b) Plan-1 - Actor Delivers 1 Package from Factory A and 1 from B:



(c) Plan-2 - Actor Delivers Both the Packages from Factory A, Helping Observer-C and Thwarting Observer-X:

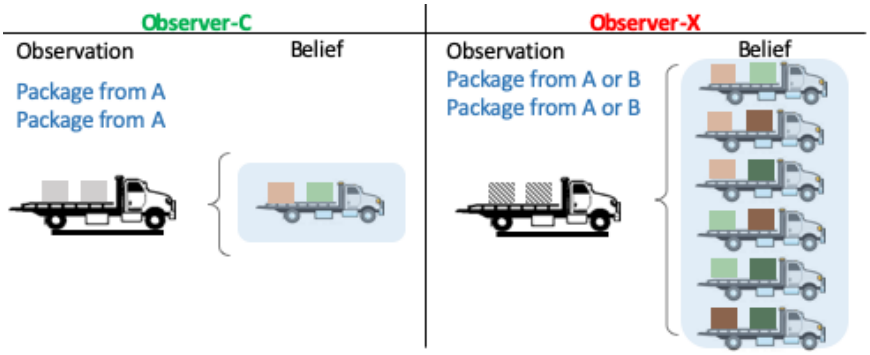


Figure 7.1: The Differences in Belief Updates Induced by the Same Plan for Two Observers with Noisy Sensors. Here Observer-X is Adversarial and Observer-C is Cooperative.

Example

Consider the example in Figure 7.1, where a truck agent has to deliver 2 packages. There are 4 packages as shown in Figure 7.1a: two at factory A and two at factory B. Therefore, there are 6 candidate goals in total (i.e., all combinations of cardinality 2). Let there be two observers who know the initial inventory at each factory, but do not know which two packages will be delivered by the agent. Each observer has different sensors (or sources of information) about the truck's activities, such that, observer-C (cooperative observer) can differentiate between packages loaded from A and B (but not the individual package in a factory), whereas observer-X (adversarial observer) cannot differentiate between any of the packages. The truck agent wants to convey its true goal to observer-C but obfuscate it to observer-X.

Consider plan-1 in Figure 7.1b, where the truck agent loads one package each from A and B. Here, observer-C cannot distinguish the packages (there are 4 possible goals that satisfy the observations). Now, consider plan-2 in Figure 7.1c, where the truck agent loads both the packages from A (or loads both the packages from B). Here, observer-C can infer the identity of the loaded packages (there is only 1 possible goal), while observer-X still does not know the identity of the packages (there are 6 possible goals). Therefore, plan-2 makes the true goal legible to observer-C while obfuscating the true goal with all the possible goals for observer-X.

7.1 MO-COPP

This problem setting involves an actor (robot) and two observers, where one observer is adversarial towards the actor whereas the other is cooperative. We assume that the actor has full observability of its activities and knows the sensor models used by the observers. When the actor takes an action and reaches a new state, each ob-

server receives an observation. If the observers have different sensors, they will receive different observations for the same activity. After obtaining the observations, each observer updates its belief. The actor leverages the known limits in the observers' sensors to control the observability of multiple observers in the environment simultaneously. Our formal framework can be extended easily to multiple adversarial and cooperative observers, each with its own set of noisy sensors. We assume that the true goal of the actor is not known to either of the observers and that the actor may use sub-optimal cost plans to hide or convey information about goals. Therefore, our objective is to develop algorithms that would allow the actor to hide its true goal from the adversarial observer by maximizing the number of candidate goals possible in the adversary's belief, and to convey its true goal to the cooperative observer by minimizing the number of candidate goals possible in that observer's belief.

Let $\mathbf{A}$ represent the actor, $\mathbf{C}$ represent a cooperative observer and $\mathbf{X}$ represent an adversarial observer.

Definition 36. *A **mixed-observer controlled observability planning problem** is a tuple,* MO-COPP $= \langle \Lambda, \mathcal{M}, \mathcal{G}, \{\Omega_i\}_{i\in\Lambda}, \{\mathcal{O}_i\}_{i\in\Lambda}, \{\mathcal{B}_0^i\}_{i\in\{\mathbf{X},\mathbf{C}\}} \rangle$*, where,*

- $\Lambda = \{\mathbf{A}, \mathbf{C}, \mathbf{X}\}$ *is the set of agents.*
- $\mathcal{M} = \langle \mathcal{F}, \mathcal{A}, \mathcal{I}, G_A, c_A \rangle$ *is* $\mathbf{A}$*'s planning problem.*
- $\mathcal{G} = \{G_1, G_2, \ldots, G_{n-1}, G_A\}$ *is the set of candidate goals, where* G_A *is the true goal of* $\mathbf{A}$*, which is not known to both* $\mathbf{C}$ *and* $\mathbf{X}$*.*
- Ω_i *is the set of observation symbols for agent* i*, which are emitted when* $\mathbf{A}$ *takes an action and reaches a new state. Further,* $\Omega_A = \{o^{\mathbf{A}}_{a,s} | a \in \mathcal{A}, s \in \mathcal{S}\}$*.*
- $\mathcal{O}_i : \mathcal{A} \times \mathcal{S} \to \Omega_i$ *is agent* i*'s deterministic observation function. Further,* $\mathcal{O}_{\mathbf{A}}$ *is an accurate observation function that maps each action-state pair to a unique*

observation, $\forall\ a, a' \in \mathcal{A},\ s, s' \in \mathcal{S}, a \neq a' \wedge s \neq s' : \mathcal{O}_{\mathbf{A}}(a, s) \neq \mathcal{O}_{\mathbf{A}}(a', s')$*, while* $\mathcal{O}_{\mathbf{X}}$ *and* $\mathcal{O}_{\mathbf{C}}$ *are noisy observation functions that map multiple action-state pairs to the same observation symbol.*

- $\mathcal{B}_0^i$ *is the initial belief of an observer,* $i \in \{\mathbf{X}, \mathbf{C}\}$*. The initial belief is a set of states inclusive of* $\mathcal{I}$*.*

From the above definition, we see that, while **X** and **C** can have arbitrary sensor models [1] with partial observability, **A** has full observability, due to one-to-one mapping of its observation function. The observers are aware of the planning domain of the actor and of the candidate goals, except they do not know which candidate goal is the actor's true goal, $G_{\mathbf{A}}$.

Sensor Models

We assume that every time the actor acts, each $i \in \Lambda$ receives an observation consistent with its observation function. The observation function of an observer $i \in \{\mathbf{X}, \mathbf{C}\}$ supports many-to-one mapping of $\langle a, s \rangle$ pairs to observation symbols, i.e., $\exists a, a' \in \mathcal{A}, s, s' \in \mathcal{S}, a \neq a' \wedge s \neq s' : \mathcal{O}_i(a, s) = \mathcal{O}_i(a', s')$. For an agent i, the inverse of observation function gives the set of $\langle a, s \rangle$ pairs consistent with an observation symbol $o^i \in \Omega_i$, represented as $\mathcal{O}_i^{-1}(o^i)$.

Each observer $i \in \{\mathbf{X}, \mathbf{C}\}$ maintains its own belief, which is a set of states. When **A** takes an action a in state s and reaches new state s' at time t, the observers receive observation o_t^i, and perform a belief update, $\mathcal{B}_t^i = \{\hat{s} \mid \exists \hat{a}, \bar{s}\ \ \Gamma(\bar{s}, \hat{a}) = \hat{s}; \bar{s} \in \mathcal{B}_{t-1}^i; \mathcal{O}_i(\hat{a}, \hat{s}) = o_t^i\}$. That is, the belief is updated using the previous belief and the observation received. For a plan π of horizon $\mathcal{T}$, the final belief of observer i is represented as $\mathcal{B}_{\mathcal{T}}^i$ and π is associated with a sequence of observations, $ObsSeq_i(\pi) =$

[1] Sensor model of an agent i refers to $\langle \Omega_i, \mathcal{O}_i \rangle$.

$\{o_1^i, \ldots, o_{\mathcal{T}}^i\}$ for each observer i. Note that the belief of $\mathbf{A}$ is essentially a single state due to full observability.

Incorrect Sensor Model

It is also possible that an observer's sensor model is compromised and is showing incorrect mapping, in the sense that it attributes observations to $\langle a, s \rangle$ pairs that could never truly yield those observations. More precisely, when an observer i's true (or uncompromised) observation function is $\mathcal{O}_i$, it may be mistaken, and believe that its observation function is $\mathcal{O}'_i$, and $\forall a \in \mathcal{A}, s \in \mathcal{S}, \mathcal{O}'_i(a, s) \neq \mathcal{O}_i(a, s)$. For e.g., the agent goes north but the observation function gives observation that is consistent with going east or west.

Proposition 7. *Given observer $i \in \{\mathbf{X}, \mathbf{C}\}$ has an incorrect observation function, $\mathcal{O}'_i$, when $\mathbf{A}$ executes $a_t \in \mathcal{A}$ and reaches $s_t \in \mathcal{S}$ emitting $o_t^i \in \Omega_i$ at $t \in \{1, \ldots, \mathcal{T}\}$, the belief update satisfies the following: $s_t \notin \mathcal{B}_t^i$.*

When $\mathbf{A}$ acts, $\mathcal{O}_i(a_t, s_t) = o_t^i$. By definition of incorrect sensor model, $\forall a \in \mathcal{A}, s \in \mathcal{S}, \mathcal{O}_i(a, s) \neq \mathcal{O}'_i(a, s)$. Hence, $\langle a_t, s_t \rangle \notin \mathcal{O}_i'^{-1}(o_t^i)$. Further, by definition of belief update, $s_t \notin \mathcal{B}_t^i$.

The above proposition states that if an observer's sensor model has an inaccurate mapping as defined above, then the true state of $\mathbf{A}$ is not part of its belief update. If $\mathbf{A}$ knows that observer i is using $\mathcal{O}'_i$, instead of $\mathcal{O}_i$, it may use it to its advantage, either to obfuscate further or to convey the true objective despite it. We focus on uncompromised sensor models for clarity in technical descriptions, but our approach extends naturally to incorrect sensor models.

7.1.1 MO-COPP *Solution*

A plan π solves a MO-COPP $= \langle \Lambda, \mathcal{M}, \mathcal{G}, \{\Omega_i\}_{i\in\Lambda}, \{\mathcal{O}_i\}_{i\in\Lambda}, \{\mathcal{B}_0^i\}_{i\in\{\mathbf{X},\mathbf{C}\}}\rangle$, iff: $\Gamma(\mathcal{I}, \pi) \models G_A$. In other words, any solution to $\mathcal{M}$ is a solution to MO-COPP. We measure the quality of a MO-COPP solution in terms of the extent to which the plan is legible to $\mathbf{C}$ and obfuscating to $\mathbf{X}$. Given a solution plan, π, let $\mathcal{G}^\pi_{\mathbf{X}} = \{G \in \mathcal{G} : \exists s \in \mathcal{B}^{\mathbf{X}}_{\mathcal{T}},\ s \models G\}$, denote the goals present in $\mathbf{X}$'s final belief and $\mathcal{G}^\pi_{\mathbf{C}} = \{G \in \mathcal{G} : \exists s \in \mathcal{B}^{\mathbf{C}}_{\mathcal{T}},\ s \models G\}$, denote the goals present in $\mathbf{C}$'s final belief. Here $|\mathcal{G}^\pi_{\mathbf{X}}|$ represents the amount of **goal obfuscation for X**, and by increasing the number of goals in $\mathcal{G}^\pi_{\mathbf{X}}$, we can increase the goal obfuscation. Similarly, $|\mathcal{G}^\pi_{\mathbf{C}}|$ represents the amount of **goal legibility for C**, and by decreasing the number of goals in $\mathcal{G}^\pi_{\mathbf{C}}$, we can increase the goal legibility. We now introduce a goal difference metric to estimate the quality of a solution plan to MO-COPP.

Definition 37. *Given a solution plan π that solves* MO-COPP*, where $|\mathcal{G}| = n$, the* ***goal difference****, of π is given by:*

$$GD(\pi) = \frac{|\mathcal{G}^\pi_{\mathbf{X}}| - |\mathcal{G}^\pi_{\mathbf{C}}|}{n-1} \tag{7.1}$$

where $n-1$ is the difference between ideal values of $\mathcal{G}^\pi_{\mathbf{X}}$ and $\mathcal{G}^\pi_{\mathbf{C}}$.

An optimal solution to MO-COPP simultaneously maximizes goal obfuscation and goal legibility. In order to maximize goal obfuscation to $\mathbf{X}$ while maximizing goal legibility to $\mathbf{C}$, we can maximize the difference between the number of goals in $\mathcal{G}^\pi_{\mathbf{X}}$ and $\mathcal{G}^\pi_{\mathbf{C}}$, as the number of goals in $\mathcal{G}^\pi_{\mathbf{X}}$ have to be higher in number while those in $\mathcal{G}^\pi_{\mathbf{C}}$ have to be lower in number. This is exactly what equation 7.1 does. Higher the goal difference, more the goal obfuscation and goal legibility. Equivalently, closer the $GD(\pi)$ value to 1, better is the plan quality. A solution plan with $GD(\pi) = 1$ is an optimal plan. E.g., in Figure 7.1c, plan-2 is an optimal plan with $GD(plan\text{-}$

$2) = \frac{6-1}{6-1} = 1$. The denominator is essential for comparing the quality of plans across different problems, with varying number of candidate goals.

Let $\hat{\pi}$ be a valid plan that is consistent with observation sequence of a solution plan, π, and the set of such plans is represented as, $ObsSeq_i^{-1}(\pi)$ for observer i. Given the observer's initial belief, there exists a valid plan that satisfies each goal in observer's final belief. That is, $\forall i \in \{\mathbf{X}, \mathbf{C}\}, G \in \mathcal{G}_i^\pi, \exists s_0^i \in \mathcal{B}_0^i,\ \hat{\pi} \in ObsSeq_i^{-1}(\pi) : \Gamma(s_0^i, \hat{\pi}) \models G$.

Proposition 8. *Given a solution plan, π, to* MO-COPP*, if $|\mathcal{G}_{\mathbf{C}}^\pi| = 1$, then $G_{\mathbf{A}} \in \mathcal{G}_{\mathbf{C}}^\pi$.*

Proof. By Definition 36, $\mathcal{I} \in \mathcal{B}_0^{\mathbf{C}}$. $\mathbf{C}$ receives a sequence of observations $\{o_1^{\mathbf{C}}, \ldots, o_{\mathcal{T}}^{\mathbf{C}}\}$. While performing belief update at any time step t, $\langle a_t, s_t \rangle \in \mathcal{O}_{\mathbf{C}}^{-1}(o_t^{\mathbf{C}})$. Therefore, $s_t \in \mathcal{B}_t^{\mathbf{C}}$. Further, solution to MO-COPP satisfies $\Gamma(\mathcal{I}, \pi) \models G_{\mathbf{A}}$, therefore $G_A \in \mathcal{G}_{\mathbf{C}}^\pi$. □

The above proposition states that when maximum goal legibility is achieved, only one goal $G_{\mathbf{A}}$ is present in $\mathbf{C}$'s final belief.

7.2 MO-COPP Plan Generation

We now present two solution approaches. In the first approach, we formulate MO-COPP as a constraint optimization problem and provide an IP encoding to solve it in $\mathcal{T}$ steps. In the second approach, we use a heuristic-guided forward search to achieve preset levels of goal obfuscation and legibility. The IP encoding provides an optimal solution for the given horizon by maximizing obfuscation while maximizing legibility, while the search algorithm generates solutions that satisfy a prespecified lower bound on the amount of goal obfuscation and goal legibility.

7.2.1 MO-COPP *as Integer Program*

In the following, we present our IP encoding.

Variables

We require the following binary variables for our encoding: **(1)** $\forall a \in \mathcal{A}, t \in \{1, \ldots, \mathcal{T}\}$, $x_{a,t}$ is an indicator variable for action a at time t, **(2)** $\forall s \in \mathcal{S}, t \in \{0, \ldots, \mathcal{T}\}$, $y_{s,t}$ is an indicator variable for state s at time t, **(3)** $\forall i \in \{\mathbf{X}, \mathbf{C}\}, o^i \in \Omega_i, t \in \{0, \ldots, \mathcal{T}\}$, $w^i_{o,t}$ is an indicator variable for observation o^i at time t, **(4)** $\forall i \in \{\mathbf{X}, \mathbf{C}\}, s \in \mathcal{S}, t \in \{0, \ldots, \mathcal{T}\}$, $b^i_{s,t}$ is an indicator variable for state s in belief $\mathcal{B}^i$ at time t, **(5)** $\forall i \in \{\mathbf{X}, \mathbf{C}\}, s \in \mathcal{S}, a \in \mathcal{A}, t \in \{0, \ldots, \mathcal{T}\}$, $h^i_{s,a,t}$ is an indicator variable for action a being applicable in state s in belief $\mathcal{B}^i$ at time t, **(6)** $\forall i \in \{\mathbf{X}, \mathbf{C}\}, G \in \mathcal{G}$, $g^i_{G,\mathcal{T}}$ is an indicator variable for a goal G present in belief $\mathcal{B}^i_{\mathcal{T}}$.

Objective Function

The IP objective function is given by:

$$maximize \sum_{G \in \mathcal{G}} g^{\mathbf{X}}_{G,\mathcal{T}} - \sum_{G \in \mathcal{G}} g^{\mathbf{C}}_{G,\mathcal{T}} \tag{7.2}$$

In equation (7.2), the first term denotes $|\mathcal{G}^{\pi}_{\mathbf{X}}|$ i.e. amount of goal obfuscation for $\mathbf{X}$, and the second term denotes $|\mathcal{G}^{\pi}_{\mathbf{C}}|$ i.e. amount of goal legibility for $\mathbf{C}$. Essentially, it finds a solution with maximum possible GD. We skip the denominator of the GD metric, as it is a constant and does not contribute to the optimization. In the first term, we maximize the goal obfuscation with respect to $\mathbf{X}$ and in the second term we maximize the goal legibility with respect to $\mathbf{C}$. This provides a single solution that achieves the maximum difference between the number of goals possible for the two observers. Note that, it would make sense to get the Pareto optimal solutions if we wanted to explore all the combinations of goals achieved for the two observers. However, that is not our objective.

Constraints

The IP constraints are written as:

$$\forall s \in \mathcal{S}, s = \mathcal{I} : \; y_{s,0} = 1 \tag{7.3}$$

$$\forall s \in \mathcal{S}, s \neq \mathcal{I} : \; y_{s,0} = 0 \tag{7.4}$$

$$\forall s \in \mathcal{S} : \sum_{G_A \in \; s} y_{s,\mathcal{T}} = 1 \tag{7.5}$$

$$\forall i \in \{\mathbf{X}, \mathbf{C}\}, s \in \mathcal{B}_0^i : \; b_{s,0}^i = 1 \tag{7.6}$$

$$\forall i \in \{\mathbf{X}, \mathbf{C}\}, s \notin \mathcal{B}_0^i : \; b_{s,0}^i = 0 \tag{7.7}$$

$$\forall i \in \{\mathbf{X}, \mathbf{C}\}, G \in \mathcal{G}, m > |\{s|\; G \in s\}| : \; m * g_{G,\mathcal{T}}^i - \sum_{G \; \in \; s} b_{s,\mathcal{T}}^i \geqslant 0 \tag{7.8}$$

$$\forall a \in \mathcal{A}, t \in \{1, \ldots, \mathcal{T}\}, \; pre_a = \{s \mid pre(a) \in s\} : \quad x_{a,t} \leqslant \sum_{s \in pre_a} y_{s,t-1} \tag{7.9}$$

$$\forall s, s' \in \mathcal{S}, t \in \{1, \ldots, \mathcal{T}\}, \; add_{s'} = \{a|\; pre(a) \in s \wedge add(a) \setminus delete(a) \in s'\},$$

$$pre_{s'} = \{s|\; pre(a) \in s \wedge add(a) \setminus delete(a) \in s'\} :$$

$$\sum_{a \in add_{s'}} x_{a,t} + \sum_{s \in pre_{s'}} y_{s,t-1} - 2 \; y_{s',t} \geqslant 0 \tag{7.10}$$

$$\forall a \in \mathcal{A}, t \in \mathcal{T}, \; pre_a = \{s \mid pre(a) \in s\}, \quad post_a = \{s' \mid add(a) \setminus delete(a) \in s'\} :$$

$$\sum_{s \in pre_a, s' \in post_a} y_{s,t-1} \; y_{s',t} = x_{a,t} \tag{7.11}$$

$$\forall i \in \{\mathbf{X}, \mathbf{C}\}, o \in \Omega_i, t \in \{1, \ldots, \mathcal{T}\} : \; w_{o,t}^i = \sum_{a, s' \in O_o^i} x_{a,t} \; y_{s',t} \tag{7.12}$$

$$\forall i \in \{\mathbf{X}, \mathbf{C}\}, s \in \mathcal{S}, t \in \{1, \ldots, \mathcal{T}\}, \; a \in add_s, add_s = \{a|\; pre(a) \in s\} :$$

$$b_{s,t-1}^i + w_{o,t}^i - h_{s,a,t}^i \leqslant 1 \tag{7.13}$$

$$\forall i \in \{\mathbf{X}, \mathbf{C}\}, s \in \mathcal{S}, o \in \Omega_i, t \in \{1, \ldots, \mathcal{T}\}, a \in add_s, add_s = \{a|\; pre(a) \in s\} :$$

$$h_{s,a,t}^i - b_{s,t-1}^i \leqslant 0 \tag{7.14}$$

$$\forall i \in \{\mathbf{X}, \mathbf{C}\}, s \in \mathcal{S}, t \in \{1, \ldots, \mathcal{T}\}, a \in add_s, s' \in post_s$$

$$add_s = \{a|\ pre(a) \in s\}, post_s = \{s'|\ add(a) \setminus delete(a) \in s'\} :$$

$$h^i_{s,a,t} - b^i_{s',t} \leqslant 0 \tag{7.15}$$

$$\forall i \in \{\mathbf{X}, \mathbf{C}\}, s \in \mathcal{S}, o \in \Omega_i, t \in \{1, \ldots, \mathcal{T}\}, a \in add_s, add_s = \{a|\ pre(a) \in s\} :$$

$$h^i_{s,a,t} - w^i_{o,t} \leqslant 0 \tag{7.16}$$

$$\forall i \in \{\mathbf{X}, \mathbf{C}\}, s, s' \in \mathcal{S}, t \in \{1, \ldots, \mathcal{T}\},$$

$$add_{s'} = \{a|\ pre(a) \in s \wedge add(a) \setminus delete(a) \in s'\},$$

$$pre_{s'} = \{s|\ pre(a) \in s \wedge add(a) \setminus delete(a) \in s'\} :$$

$$\sum_{s \in pre_{s'}, a \in add_{s'}} h^i_{s,a,t} - b^i_{s',t} \geqslant 0 \tag{7.17}$$

$$\forall t \in \{1, \ldots, \mathcal{T}\} : \sum_{a \in \mathcal{A}} x_{a,t} \leqslant 1 \tag{7.18}$$

Constraints (7.3) and (7.4) initialize the state variable for initial state, (7.5) says that a state that satisfies the true goal should be achieved in the last time step for $\mathbf{A}$. Constraints (7.6) and (7.7) initialize the initial belief variable for both the observers. Constraint (7.8) says that if a goal is satisfied in the final belief of an observer then the corresponding goal variable will be true for that observer. Constraint (7.9) through (7.11) enforce the transition function, $\Gamma(\cdot)$, on actor's state and action. Specifically, constraint (7.9) validates the applicability of an action in a state, constraint (7.10) states that for a resulting state to be true both the action and the state in which it is applied should be true, and similarly constraint (7.11) validates an action with respect to its previous state and the resulting state. Constraint (7.12) enforces the corresponding observation symbol for each observer depending on the $\langle a, s' \rangle$ pair. Constraints (7.13) through (7.17) enforce a belief update. Specifically, constraint (7.13) states that an action is not applicable in a belief state if either the belief state

or the observation is untrue. Constraint (7.14) states that an action cannot be applied in a belief state that is untrue. Constraint (7.15) states that an action cannot be true if the resulting belief state is untrue. Constraint (7.16) states that an action cannot be true if the corresponding observation is untrue. Constraint (7.17) states that a belief state is true if the sum of actions leading to it is at least 1. Constraint (7.18) ensures at most one action is possible at each time step for the actor.

Proposition 9. *The IP encoding listed above which takes time horizon $\mathcal{T}$ as input, solves a* MO-COPP $= \langle \Lambda, \mathcal{M}, \mathcal{G}, \{\Omega_i\}_{i\in\Lambda}, \{\mathcal{O}_i\}_{i\in\Lambda}, \{\mathcal{B}_0^i\}_{i\in\{\mathbf{X},\mathbf{C}\}} \rangle$ *such that, the following properties hold:*

- ***Soundness**: A solution to the IP will solve* MO-COPP.
- ***Completeness**: If there exists a plan that solves the* MO-COPP *in $\mathcal{T}$ time steps, then it will be a feasible solution for the IP encoding.*
- ***Optimality**: An optimal solution to the IP encoding will be a plan that solves the* MO-COPP *with an optimal value of GD given the time horizon, $\mathcal{T}$.*

Since a solution to the IP, π_{IP}, satisfies (7.4), it solves MO-COPP. If there exists a plan $\pi_{\mathcal{T}}$ of $\mathcal{T}$ time steps that solves MO-COPP, $\Gamma(\mathcal{I}, \pi_{\mathcal{T}}) \models G_A$, then $\pi_{\mathcal{T}}$ will satisfy the constraints (7.4), (7.9)-(7.11) and (7.18). The IP encoding uses the numerator of GD metric as its objective function. An optimal solution to the IP encoding, π^*_{IP}, is an optimal plan to MO-COPP given plans of $\mathcal{T}$ time steps.

Modulating Actor's Behavior

The objective function presented in (7.2) trades off goal obfuscation with goal legibility for the observers. However, the actor can ensure a predefined level of goal

obfuscation (say obfuscate with *at least* k candidate goals), using the following constraint:

$$\sum_{G \in \mathcal{G}} g^{\mathbf{X}}_{G,\mathcal{T}} \geqslant k, \quad s.t.\ 1 \leqslant k \leqslant |\mathcal{G}| \tag{7.19}$$

Similarly, to ensure predefined level of goal legibility with respect to certain number of candidate goals (say legible with *at most* j goals), the following constraint can be added:

$$\sum_{G \in \mathcal{G}} g^{\mathbf{C}}_{G,\mathcal{T}} \leqslant j, \quad s.t.\ 1 \leqslant j \leqslant |\mathcal{G}| \tag{7.20}$$

These kind of constraints allow the actor to filter out solutions that do not satisfy minimum bound for goal obfuscation and goal legibility. The actor can improve the robustness of the plans generated by using these constraints.

7.2.2 Search Algorithm

In this section, we show that it is possible to leverage search techniques that address goal obfuscation and goal legibility in isolation to solve MO-COPP. We adapt Algorithm 3 introduced in Chapter 4 to address goal obfuscation and goal legibility simultaneously to two different observers. We specify bounds on the amount of goal obfuscation and goal legibility desired, similar to the ones seen in the IP: obfuscate with at least k goals, make it legible with at most j goals. These bounds, $\Phi = \langle \Phi_{\mathbf{X}}, \Phi_{\mathbf{C}} \rangle$, are given as input to the search algorithm. Note that, in order to guide the search with the heuristic described below, the candidate goals need to be chosen beforehand.

Each search node maintains the associated beliefs for both observers. The *approx* function generates an approximate belief, b^i_Δ, of size Δ (i.e. cardinality of b^i_Δ is Δ).

Algorithm 7 Heuristic-Guided Search

Initialize *open*, *closed* and *temp* lists; $\Delta = 1$
$\langle b^X_\Delta, b^C_\Delta \rangle \leftarrow \text{approx}(\mathcal{I}, \mathcal{B}^X_0, \mathcal{B}^C_0)$
$open.push(\mathcal{I}, \langle b^X_\Delta, b^C_\Delta \rangle, \langle \mathcal{B}^X_0, \mathcal{B}^C_0 \rangle, priority = 0)$
while $\Delta \leqslant |\mathcal{S}|$ **do**
    while $open \neq \emptyset$ **do**
        $s, \langle b^X_\Delta, b^C_\Delta \rangle, \langle \mathcal{B}^X, \mathcal{B}^C \rangle, h_{node} \leftarrow open.pop()$
        if $|b^X_\Delta| > \Delta$ or $|b^C_\Delta| > \Delta$ **then**
            $temp.push(s, \langle b^X_\Delta, b^C_\Delta \rangle, \langle \mathcal{B}^X, \mathcal{B}^C \rangle, h_{node})$
            continue
        end if
        add $\langle b^X_\Delta, b^C_\Delta \rangle$ to *closed*
        if $s \models G_A$ and $\mathcal{B}^X \models \Phi_X$ and $\mathcal{B}^C \models \Phi_C$ **then**
            return $\pi, ObsSeq_X(\pi), ObsSeq_C(\pi)$
        end if
        for $s' \in$successors(s) **do**
            $o^X \leftarrow \mathcal{O}_X(a, s')$; $o^C \leftarrow \mathcal{O}_C(a, s')$
            $\widehat{\mathcal{B}^X} = \text{Update}(\mathcal{B}^X, o^X); \widehat{\mathcal{B}^C} = \text{Update}(\mathcal{B}^C, o^C)$
            $\langle \widehat{b^X_\Delta}, \widehat{b^C_\Delta} \rangle \leftarrow \text{approx}(s', \widehat{\mathcal{B}^X}, \widehat{\mathcal{B}^C})$
            $h_{node} \leftarrow h_{G_A}(s') + h_{\mathcal{G}_{k-1}}(\widehat{\mathcal{B}^X}) - h_{\mathcal{G}_{\mathcal{G}-j}}(\widehat{\mathcal{B}^C})$
            add new node to *open* if $\langle \widehat{b^X_\Delta}, \widehat{b^C_\Delta} \rangle$ not in *closed*
        end for
    end while
    increment Δ; copy items from *temp* to *open*; empty *temp*
end while

b^i_Δ is always inclusive of the true state of the actor, this is because the actor can only take actions that are consistent with its true state. If all such $\Delta-$sized beliefs are explored then b^i_Δ of $\Delta+1$ size is computed, and this node gets put in the temporary list and is explored in the next outer iteration when Δ has been incremented. For each Δ, all Δ-sized unique combinations of belief (that include the actual state of the actor) are explored. This allows systematic and complete exploration of multiple paths to a given search node. The inner iteration performs heuristic guided forward search (we use greedy best first search) to find a plan while tracking at most Δ states in each b^i_Δ. In the inner loop, the node expansion is guided by (1) customized heuristic function, which computes value of the node based on true goal and belief constraints given by Φ for the observers, and (2) goal test, which checks for satisfaction of true goal and satisfaction of the belief constraints given by Φ. The algorithm stops either when a solution is found or when all the Δ iterations have been explored.

Proposition 10. *The search algorithm listed above which takes goal-constraints Φ as input, solves a* MO-COPP $= \langle \Lambda, \mathcal{M}, \mathcal{G}, \{\Omega_i\}_{i\in\Lambda}, \{\mathcal{O}_i\}_{i\in\Lambda}, \{\mathcal{B}^i_0\}_{i\in\{\mathbf{X},\mathbf{C}\}}\rangle$ *such that, the following properties hold:*

- ***Soundness*** *Any solution to the search algorithm is a plan that solves the* MO-COPP*.*
- ***Completeness*** *If there exists a plan that solves* MO-COPP *given goal-constraints Φ, it will be found by the search.*

A solution to the search algorithm solves MO-COPP, since the goal test ensures the true state of $\mathbf{A}$ satisfies G_A. The search algorithm necessarily terminates in $|\mathcal{S}|$ iterations of the Δ parameter. The Δ parameter allows systematic exploration of unique Δ-sized combinations of belief, starting with $\Delta = 1$ until a solution is found or the solution space is explored, $\Delta = |\mathcal{S}|$. The goal test checks for satisfaction of

constraints in Φ. Hence, a plan that solves MO-COPP given Φ will be found by the search algorithm.

Property In both the solution approaches, we can assert a lower bound on the extent of goal obfuscation and goal legibility for a MO-COPP solution plan. In IP, we can specify the aforementioned goal constraints to assert this minimum value, while in the search, the goal tests allow us to assert it. By setting k, j to desired values, we can eliminate solutions with low GD score. This affords the following guarantee:

Proposition 11. *Let* $\mathbf{X}$ *and* $\mathbf{C}$ *be perfectly rational adversarial and cooperative observers respectively. Given a* MO-COPP $= \langle \Lambda, \mathcal{M}, \mathcal{G}, \{\Omega_i\}_{i\in\Lambda}, \{\mathcal{O}_i\}_{i\in\Lambda}, \{\mathcal{B}_0^i\}_{i\in\{\mathbf{X},\mathbf{C}\}}\rangle$ *with equally probably goals,* $|\mathcal{G}| = n$*, and goal constraints of at least* k *goal obfuscation for* $\mathbf{X}$ *and at most* j *goal legibility for* $\mathbf{C}$*, then a solution plan,* π*, gives the following guarantees:*

1. $\mathbf{X}$ *can infer* $G_{\mathbf{A}}$ *with probability* $\leqslant 1/k$*,*
2. $\mathbf{C}$ *can infer* $G_{\mathbf{A}}$ *with probability* $\geqslant 1/j$*, and*
3. *Goal difference metric,* $GD(\pi) \geqslant \frac{k-j}{n-1}$

Given $|\mathcal{G}^{\pi}_{\mathbf{X}}| \geqslant k$, X can infer $G_{\mathbf{A}}$ with probability $\leqslant 1/k$. Similarly, given $|\mathcal{G}^{\pi}_{\mathbf{C}}| \leqslant j$, C can infer $G_{\mathbf{A}}$ with probability $\geqslant 1/j$. Also, similarly, $GD(\pi) \geqslant \frac{k-j}{n-1}$. The above proposition states that, based on the observation equivalence there is no additional information revealed about the actor's true goal. Therefore, we can derive goal detection upper bound for $\mathbf{X}$ and lower bound for $\mathbf{C}$. Also this allows us to derive a lower bound on the plan quality.

Exploiting Adversarial Observer's Incorrect Sensor Model

The actor can leverage $\mathbf{X}$'s incorrect sensor model (if it knows that $\mathcal{O}'_{\mathbf{X}}$ is being used, instead of $\mathcal{O}_{\mathbf{X}}$) to ensure that the true goal never appears in $\mathbf{X}$'s final belief. This can be achieved through both the solution approaches. For the IP encoding, an additional constraint $g^{\mathbf{X}}_{G_A,\mathcal{T}} = 0$ can be introduced to ensure the absence of the true goal from $\mathbf{X}$'s belief. Similarly, for the search algorithm, the goal test of the search can be modified to additionally check for absence of G_A in $\mathbf{X}$'s final belief.

Proposition 12. *Given a* MO-COPP $= \langle \Lambda, \mathcal{M}, \mathcal{G}, \{\Omega_i\}_{i\in\Lambda}, \{\mathcal{O}_i\}_{i\in\Lambda}, \{\mathcal{B}^i_0\}_{i\in\{\mathbf{X},\mathbf{C}\}}\rangle$ *where,* $\mathbf{X}$ *has an incorrect observation function,* $\mathcal{O}'_{\mathbf{X}}$*, the actor can compute a solution plan,* π*, such that,* $G_{\mathbf{A}} \notin \mathcal{G}^{\pi}_{\mathbf{X}}$.

Proof. Given the incorrect sensor model, $\mathcal{O}'_{\mathbf{X}}$, from Proposition 7, it follows that, at any time step, t, $s_t \notin \mathcal{B}^{\mathbf{X}}_t$. Also, although π is valid solution to MO-COPP that satisfies $\Gamma(\mathcal{I}, \pi) \models G_{\mathbf{A}}$, in time step $\mathcal{T}$, $G_A \notin \mathcal{B}^{\mathbf{X}}_{\mathcal{T}}$. Therefore, a valid solution plan, π can satisfy the constraints that ensure $G_A \notin \mathcal{G}^{\pi}_{\mathbf{X}}$. □

The above proposition states that the incorrect sensor model can be used by the actor to compute a solution that ensures the absence of true goal from the adversarial observer's belief. In a similar fashion, if a cooperative observer has an incorrect sensor model, by using similar constraint/goal test condition, the actor can ensure the presence of true goal for a cooperative observer. Our formulation and solution approaches allow the actor to flexibly control its activities to ensure the presence or absence of desired candidate goals.

7.3 Empirical Evaluation of MO-COPP Solutions

In this section, we evaluate the performance of our approaches against a baseline and discuss the relative strengths of both the approaches using 6 domains. We also

compare the quality of the solutions generated by our approach against an approach that assumes either entirely adversarial or entirely cooperative observers.

7.3.1 Domain Setup

For the evaluation, we look at 6 domains: Gridworld, Box Pushing, Recycling Robot, Blocksworld, Logistics and Driverlog. In Gridworld, the actor can move in cardinal directions. $\mathcal{O}_{\mathbf{X}}$, consists of two observations, *vertical* for movement along N or S and *horizontal* otherwise. $\mathcal{O}_{\mathbf{C}}$ consists of symbols, *north-east* for N or E and *south-west* otherwise. In Box Pushing [56], the actor's goal is to face the box and push it to the topmost row. The actions available are: *move-forward, stay, turn-left, turn-right*. $\mathcal{O}_{\mathbf{X}}$ consists of two symbols, *turn* when the agent performs turning actions and *move*, otherwise. $\mathcal{O}_{\mathbf{C}}$ consists of *move-right* when turning right or moving and *leftwards* otherwise. In Recycling Robot [94], the actor's objective is to collect cans and put them in recycling bins. The actions available are: *N, S, E, W, pick-up, drop, charge, stay*. $\mathcal{O}_{\mathbf{X}}$ consists of four symbols, *horizontal* when moving E or W, *vertical* when moving N or S, *using-gripper* when picking or dropping and *charging* otherwise. $\mathcal{O}_{\mathbf{C}}$ consists of *north-east*, *south-west*, *charging-picking*, and *staying-dropping*. For the three IPC domains, we use the lifted action names as observations for adversarial observer and lifted action names with objects as observations for cooperative observer. For example, given Blocksworld with 4 blocks *a, b, c, d*, for $\mathbf{C}$, we can have *stack-a-b* to represent stacking of *a* or *b* on any other block and similarly *stack-c-d*. This is done for all the lifted actions. We produced similar observations for Logistics and Driverlog. In modeling the observations, we use the notion that in realistic scenarios, it's likely that the actor may provide the ally with tools/information to infer observations more clearly.

We implemented our IP encoding using Gurobi optimizer [38]. We implemented

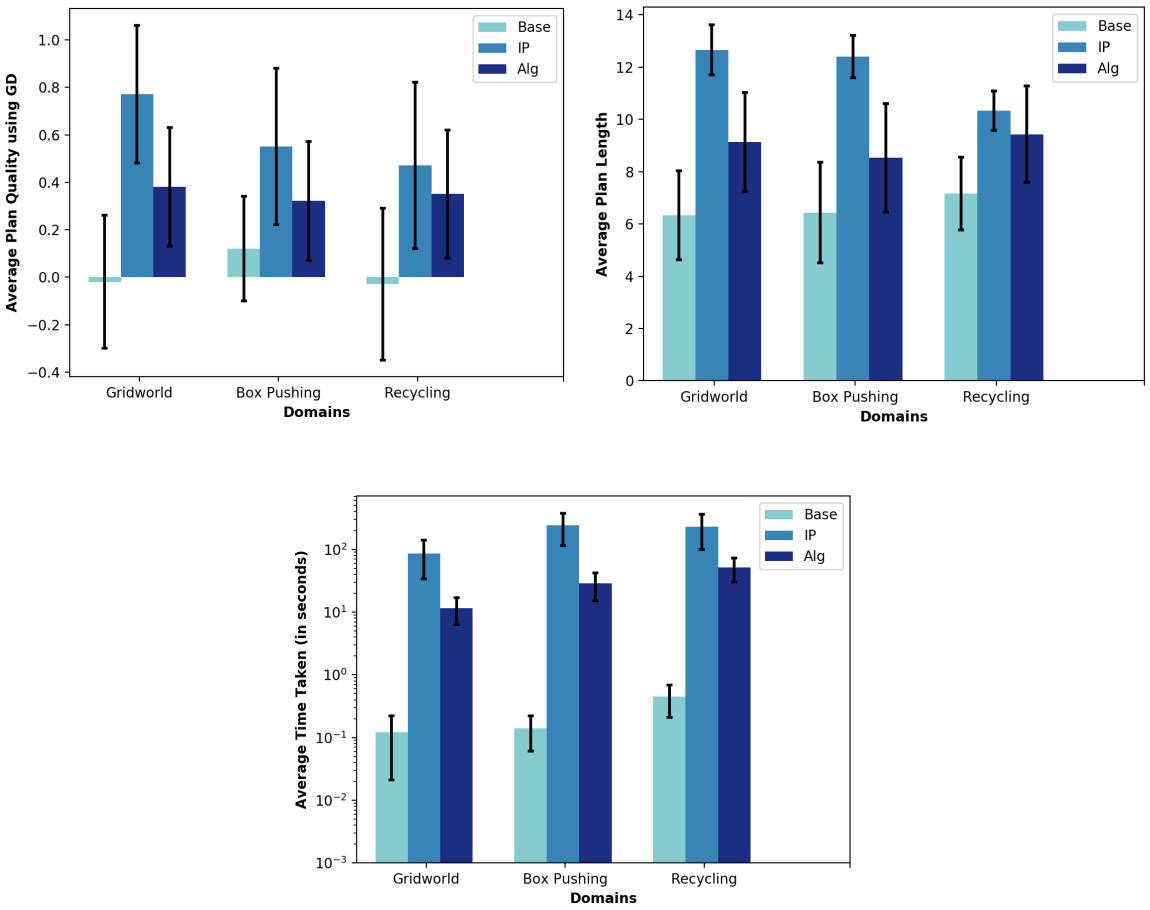


Figure 7.2: Comparison of Average and Standard Deviation for Goal Difference (GD), Plan Length and Run Time Using a Baseline Planner, IP Planner and Heuristic-guided Planner over Three Domains.

the heuristic-guided search using the STRIPS planner Pyperplan [1] with hsa [53] heuristic. We used hsa heuristic because it gave better performance. For baseline planner, we used greedy best first search with hsa heuristic. We ran our experiments on 3.5 GHz Intel Core i7 processor with 16 GB RAM. We used Gridworld of size 8x8, Box Pushing of size 5x5 with a single box, Recycling Robot of size 4x4 with a can and battery of 5 levels, Blocksworld with 5 blocks, and Logistics and Driverlog had goals with 4 facts each. For each domain, we generated 30 problems with random

initial state goals. For the first 3 domains, we created 3 goals per problem, and for the next 3 domains, 5 goals per problem.

Relative Strengths

We report the averages and standard deviation of GD, plan length and time taken for baseline, IP and search algorithm in Figure 7.2. For the baseline, we computed plans considering only $G_{\mathbf{A}}$. For the IP, we set the plan horizon to 12 for Gridworld, 12 for Box-Pushing and 10 for Recycling-Robot. If solutions were not found for that horizon, we incremented the horizon by 2. The IP did not run for the complex IPC domains, so we only display results for the first three domains. For the search, we set the minimum number of goals for $\mathbf{X}$ to 2 (at least 2 goals), and maximum number of goals for $\mathbf{C}$ to 2 (at most 2 goals). For all three algorithms, the GD is calculated by counting the total number of goals present in the observers' beliefs. We tested for statistical significance of the results by performing independent measures ANOVA to reject the null hypothesis that the three algorithms are the same and that the differences are due to any randomness in the experiments (e.g., the randomly chosen goals). For Gridworld, we found that the p-value is less than 0.00001 for GD, as well as for plan length and run time considering the results of all 3 algorithms. This is true for Box-Pushing and for Recycling-Robot as well. All the results are significant at $p < 0.05$.

The IP approach has several advantages. Firstly, it produces optimal solutions given a time horizon for the MO-COPP problem. Secondly, it provides a lot of flexibility: it automatically chooses the best candidate goals to be added to or removed from the final beliefs of the observers. Also, if we want to specifically add or remove a particular goal from the observer's final belief, it is easy to add the necessary constraint. These advantages were evident in the results: IP has higher GD for all the 3

domains. On the other hand, the search algorithm is faster and generates satisficing solutions that meet the goal constraints: the average time for search is consistently lower than IP for all the domains. Also the search solutions are shorter in length than those of the IP. The baseline although fastest (satisficing solution to a single goal) produces worst plan quality (GD). Additionally, the search can run more complex problems. The average and standard deviation GD for the IPC domains is reported in Figure 7.3.

Domain	Metric	GD
Blocksworld	Avg	0.26
	Std	0.19
Logistics	Avg	0.24
	Std	0.21
Driverlog	Avg	0.36
	Std	0.26

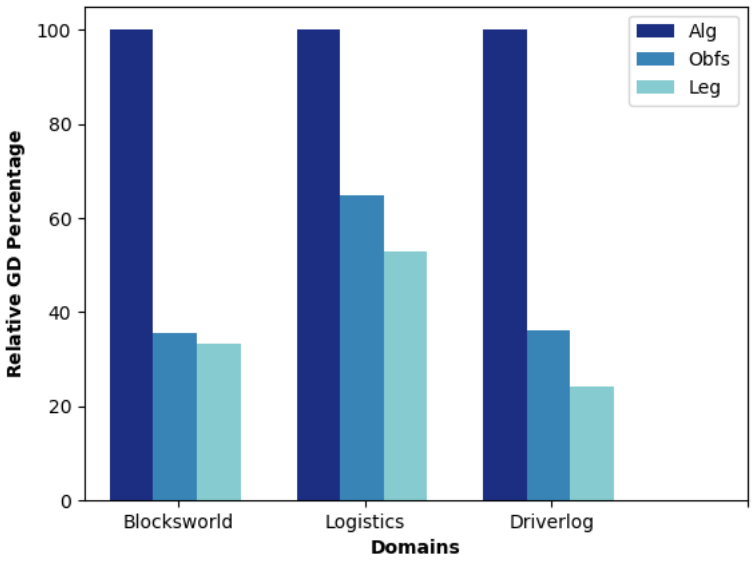


Figure 7.3: (a) Table Shows the Average and Standard Deviation GD for IPC Domains. (b) Graph Shows Relative GD Between Our Algorithm and Approaches That Achieve Obfuscation/Legibility in Isolation.

Comparison Against Other Approaches

Here, we report the GD metric for IPC domains and show the relative performance of our algorithm against approaches that assume entirely adversarial or entirely cooperative observers using IPC domains. We used k-ambiguous and j-legible algorithms

from [59] for achieving obfuscation and legibility in isolation as baseline. We compute the GDs for the baseline goal obfuscation and goal legibility by allowing minimum constraint for the other case. That is, when running goal obfuscation algorithm, the minimum constraint is to be legible with respect to at most 5 goals. Similarly, when using goal legibility algorithm, the minimum constraint is to be obfuscating with respect to at least 1 goal. We set a timeout of 20 minutes, and altogether 7 problems timed out (3 from Logistics, 4 from Driverlog) out of 90 problems. Here we set k and j values to 3. In Figure 7.3, we report the relative GD percentages for solutions that achieve goal obfuscation and goal legibility in isolation with respect to GD of the solutions produced by our algorithm. The GD is computed by counting the number of goals in each observer's belief.

From Figure 7.3, we can see that, our approach consistently outperforms the obfuscation and legibility algorithms with respect to the plan quality of the solutions (GD). This is because, as stated in Proposition 11, our approach makes sure that each solution achieves a minimum amount of GD. In this case, the minimum is 0 (since k and j values are set to 3). This ensures that our search algorithm does not output solutions with $GD < 0$, which was not the case for the other two approaches, as is seen from the relative GD percentages. This evaluation shows that the existing approaches that address obfuscation and legibility in isolation are not sufficient to produce good quality solutions to MO-COPP.

7.4 Concluding Remarks

In this chapter, we discussed the generalization of the controlled observability planning framework, namely MO-COPP, that supports both cooperative as well as adversarial observers. We presented two solution approaches to solve this problem: the first approach solves the problem optimally by framing it as a constraint optimization

problem and solves using an IP encoding, while the second approach leverages the COPP algorithm (Algorithm 3) to simultaneously balance goal obfuscation and goal legibility. We demonstrated the effectiveness of both the approaches using 6 domains.

Chapter 8

PLANNING FOR ASSISTIVE BEHAVIOR

So far in this thesis, the problem settings have assumed that the human observer is passive and that she only observes the robot to infer its activities. In a more realistic scenario, the human may be observing the robot's activities, to understand if they interfere with her own goals and plans. Therefore, to accommodate such a setting, we will extend the COPP formulation from a single actor (active robot - passive human observer) setting to a multi-actor (active robot and active human observer) setting, which we refer to as Multi-Agent Controlled Observability Planning Problem (MA-COPP). In this chapter, we will consider a scenario where the robot is playing the role of a proactive assistant.

While assisting the humans may be tricky for an AI agent even when the humans explicitly request for assistance, it is even more challenging for the agent to provide the assistance when it has to do it proactively. Not only does it have to reason over the human's goals to synthesize an assistive behavior that reduces the human's costs, but it also has to make sure that the assistance it provides can be recognized by the human, who may not be expecting it. Like the proverbial justice, *proactive assistance should not only be provided, but should be seen to be provided.* This further becomes challenging in environments where the human may have partial observability of the AI agent's activities. The agent thus needs to synthesize communicative behaviors – be they purely epistemic (speech acts) or ontic actions with epistemic effects – which allow the human to recognize the assistance. This requires it to control the human's observability by reasoning over her belief states.

This work specifically looks at the problem of providing proactive assistance to a

human in an environment where the AI agent and the human coexist, and have partial observability of each other's activities. There are several real-world workspaces like factory floors, warehouses, restaurants, nursing homes for elderly, disaster response areas, etc., where this problem of providing proactive task assistance to the involved humans is important. Our formulation considers a scenario where the AI agent is aware of the tasks being allocated to the human by the ecosystem and may also know the rules and protocols of the ecosystem. We assume that the agent has access to an input that captures the human's planning process for her goals. For instance, prior works that study the problem of action model acquisition [104; 103] can be used to derive the human's planning process. This allows it to synthesize assistive plans and to reason over the impact of those plan on the human's goals and plans. This leads us to the first principle: **(1)** *A proactive assistant's behavior should only decrease the human's optimal cost towards her goal.*

Further, since the agent is providing proactive assistance, it is essential for it to ensure that the human recognizes the assistance and modifies her original plan towards her goal. Additionally, our formulation accommodates environments where both the human and the AI agent may not have full visibility of each other's activities. For instance, the human may not know what activities were performed by a robot in another room. Therefore, it should be able to take into account the human's perception limitations as well as the possible ways of communicating necessary information to her. This leads us to the second principle: **(2)** *A proactive assistant should make the human aware of the potential reduction in her cost as a result of its assistance.*

Lastly, it is important to capture the cost of the assistance to the human. For instance, if the human has to wait for a really long time for the agent to provide assistance, then the human may instead prefer to work by herself. Since, the human is actively involved in the overall plan, it is not only necessary to reduce the human's

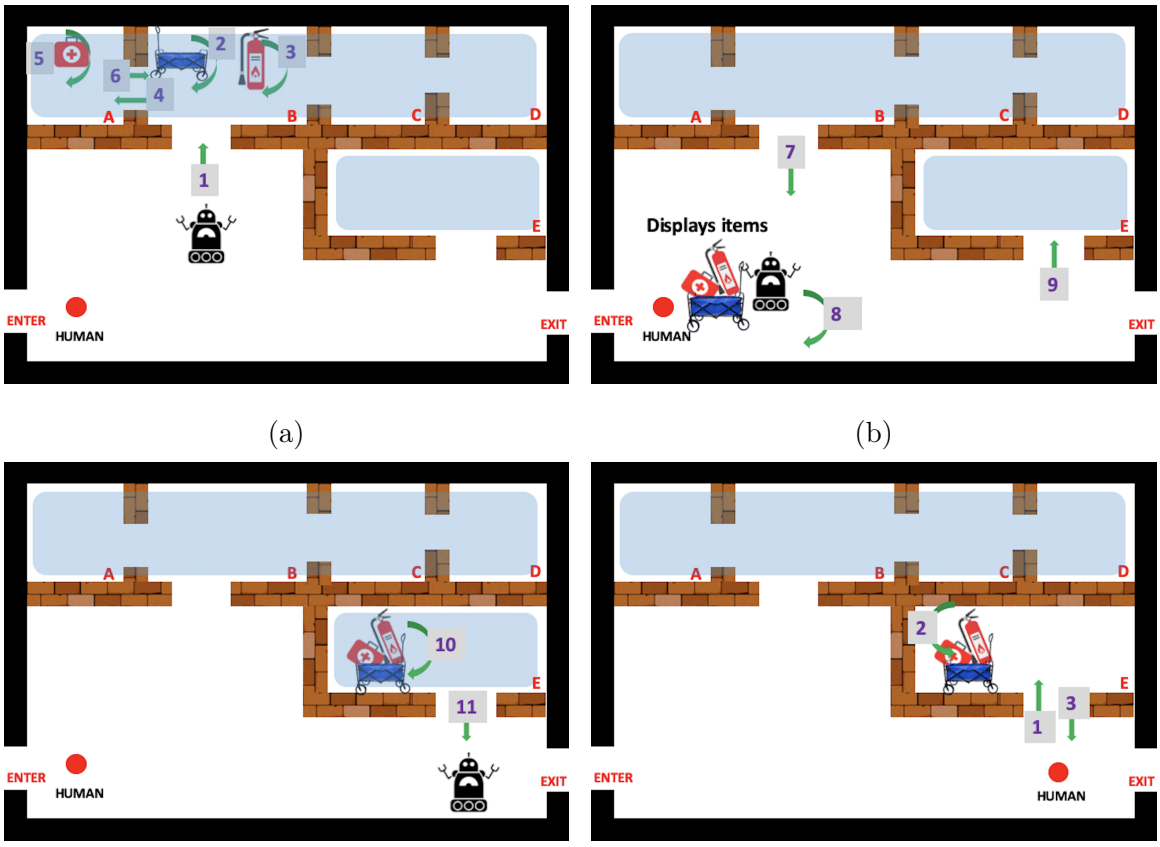


Figure 8.1: Illustration of an Assistive Joint Plan in Urban Search and Rescue Domain. (a) the Robot Collects Items Required for a Side Goal (Fire Extinguisher) and Human's Goal (Medkit) in a Wagon, (b) Makes the Human Aware of the Items It Is Carrying by Showing Them, (c) Leaves the Wagon in Room E. (d) the Human Collects the Medkit from Room E to Accomplish Her Goal.

cost to her goal, but also to reduce her overall effort resulting from processing the agent's behavior. Therefore, **(3)** *A proactive assistant should optimize for the overall cost incurred by the human in terms of the time taken (or resources needed) to participate in the overall plan.* Together these principles guide our proactive assistant. In the following sections, we propose a Monte Carlo Tree Search (MCTS) based solution that modulates the human's belief by either communicating necessary information or limiting irrelevant information (i.e. by controlling human's observability) to communicate the potential cost reduction to the human. We perform an empirical evaluation and a user study to assess the utility of our approach.

Example

Let's consider a concrete example in an urban search and rescue domain. Here a human commander and a robot are operating on a floor as shown in the Figure 8.1a. The human's task is to find a medkit on this floor. She has access to the floor map but does not know where the medkit is. Hence, her cost for accomplishing the task is very high (she has to search each room on the floor). The robot is aware of the task allocated to the human. It is also working on a non-urgent side goal of dropping a fire extinguisher to room E (as shown in Figure 8.1c). The robot who is already operating on that floor has more information about the locations of the items and is capable of assisting the human. However, since the assistance is being provided proactively, it is important for the robot to ensure that the human can recognize how the assistance optimizes her task. The robot assists as detailed in Figures 8.1a to 8.1c, where it first collects the medkit in its wagon, and then it performs an action to display the contents of the wagon to the human and then transports it to room E. This allows the human to form a belief that the medkit is in room E. She can now optimize her goal, as shown in Figure 8.1d.

8.1 Related Work

The problem of synthesizing a proactive assistant is directly connected to the prior literature on modeling assistive agents. Starting from the seminal work on SharedPlan theory [37] which discussed the notion of "intentions-to" and "intentions-that" constructs to more recent work on the development of a theory of assistance, there has been a lot of research in this general direction. In the SharedPlan theory [37], the "intentions-that" constructs refer to the acts that are performed by an agent as a responsibility towards other agents. In our framework, the AI agent's assistive actions towards the human fall under this category. In some of the more recent works [33; 75] the emphasis has been on learning the human's goal first by performing information gathering actions and following those with assistive actions towards the goal. We build on these works by assuming the human's goal is known beforehand and emphasize on how the agent can ensure the communication of its proactive assistance. Further, work by [100] delineates desired properties like pertinence, competency, etc., of a proactive assistant and proposes an operational framework. Our concrete guidelines can be traced back to these general guidelines. Other works [46; 95] have studied in a collaborative setting with predefined roles for agents, whether an agent should help others or not. In our case, the agent stops itself from assisting proactively only when the assistance puts the human in a worse-off situation.

More recently, the idea of planning for stigmergic collaboration in human-agent cohabitation scenarios [13; 10] has been explored. However, in these works, the agent does not reason over human's awareness of the assistance, in addition human's partial observability of the agent's actions is not considered. Further, unlike some decision support systems [85] that tend to be proactively assistive at planning time by providing plan suggestions, our system provides proactive task assistance at execution time.

Assistance at execution time has additional challenges of managing human's observability. Besides managing human's observability, sometimes it might be necessary to manage her attention. For instance, despite the agent's attempt at making the human aware of the assistance, she may voluntarily or involuntarily be inattentive, thereby invalidating agent's efforts. This aspect of attention management [45] has been studied in the literature. However, we do not consider the problem of human's attention management here. Lastly, here we borrow the notion of controlled observability from the earlier chapters [59; 60]. However, we show in the upcoming sections that both legibility and obfuscation can be used to communicate assistance.

8.2 MA-COPP

We consider two actors $\mathbf{R}$ (say, a robot) and $\mathbf{H}$ (say, a human). The objective of $\mathbf{R}$ is to proactively provide task level assistance to $\mathbf{H}$ at execution time. As mentioned before, by using $\mathbf{H}$'s decision algorithm as an input to our system, we can simulate $\mathbf{H}$'s plans.

In our setting, $\mathbf{R}$ is aware of the both its own model and $\mathbf{H}$'s model. Whereas, $\mathbf{H}$ is only aware of its own model. Both the agents have full observability of their own activities. However, they both have partial observability of certain actions performed by the other agent. $\mathbf{R}$ is also aware of $\mathbf{H}$'s perception limitations of its actions [1] and is capable of choosing among multiple actions to modulate $\mathbf{H}$'s observability of its actions. We call this framework as *multi-agent controlled observability planning problem* (or MA-COPP).

Definition 38. *A **multi-agent controlled observability planning problem** is a tuple,* MA-COPP $= \langle \mathcal{M}^{\mathbf{H}}, \mathcal{D}^{\mathbf{R}}, \Omega^{\mathbf{H}}, \mathcal{O}^{\mathbf{H}} \rangle$,

[1] We consider a single-interaction assistive setting and therefore do not model $\mathbf{R}$'s partial observability of $\mathbf{H}$'s actions.

- $\mathcal{M}^{\mathbf{H}} = \langle \mathcal{F}, \mathcal{A}^{\mathbf{H}}, \mathcal{B}_0, G^{\mathbf{H}}, c^{\mathbf{H}} \rangle$ *is* $\mathbf{H}$*'s planning problem.* $\mathcal{B}_0$ *is its initial belief, which is a set of states inclusive of actual initial state* $\mathcal{I}$*.*

- $\mathcal{D}^{\mathbf{R}} = \langle \mathcal{F}, \mathcal{A}^{\mathbf{R}}, \mathcal{I}, c^{\mathbf{R}} \rangle$ *is* $\mathbf{R}$*'s action model.* $\mathbf{R}$ *has full observability of its actions and states.*

- $\Omega^{\mathbf{H}}$ *is the set of observation symbols received by* $\mathbf{H}$*, when it acts or when* $\mathbf{R}$ *acts.*

- $\mathcal{O}^{\mathbf{H}} : \mathcal{A}^{\mathbf{R}} \cup \mathcal{A}^{\mathbf{H}} \times \mathcal{S} \rightarrow \Omega^{\mathbf{H}}$ *is* $\mathbf{H}$*'s observation function. Further,* $\exists a, a' \in \mathcal{A}^{\mathbf{R}},\ s, s' \in \mathcal{S}, a \neq a' \wedge s \neq s' : \mathcal{O}^{\mathbf{H}}(a, s) = \mathcal{O}^{\mathbf{H}}(a', s')$*, i.e.,* $\mathcal{O}^{\mathbf{H}}$ *gives coarse-grained observations for at least some actions of* $\mathbf{R}$*, making some of* $\mathbf{R}$*'s actions seem indistinguishable.*

From the definition of MA-COPP, we can see that although both the agents have independent action models, action costs, they share the same state space. Moreover, **H**'s initial belief consists of **R**'s initial state. To select a specific behavior that modulates **H**'s information in the environment, **R** requires access to **H**'s prior knowledge, its perception limitations as well as its task. In the running example, this involves modeling the fact that the human does not know the location of the items, as well as that she cannot see the actions performed in other rooms, and that her goal is to find a medkit on that floor.

In MA-COPP, **R** executes an assistive behavior from the initial state followed by **H**'s execution from the updated belief towards its goal. For instance, in the running example, the robot displays the wagon and leaves it in room E followed by the human commander's execution of her plan. Due to **H**'s partial observability of **R**'s actions, it operates in a belief space. For instance, the human didn't know the contents of the wagon before they are displayed. In solving MA-COPP, the challenge lies in choosing

the right amount of information to reveal to $\mathbf{H}$. $\mathbf{R}$ can select actions that reveal the missing information to $\mathbf{H}$. It can also select actions that hide away unnecessary complexities from $\mathbf{H}$. Therefore, in solving MA-COPP, $\mathbf{R}$ has to carefully choose what information to reveal versus what to hide from $\mathbf{H}$.

8.2.1 Robot Modeling of Human's Belief Update

After an action $a \in \mathcal{A}^{\mathbf{R}} \cup \mathcal{A}^{\mathbf{H}}$ changes the current state of world resulting in a new state $s \in \mathcal{S}$, $\mathbf{R}$ simulates $\mathbf{H}$'s belief update by using the observation $\mathcal{O}^{\mathbf{H}}(a, s)$ emitted by $\mathbf{H}$'s sensor model. The definition of $\mathbf{H}$'s sensor model also allows for actions with null observations. That is, a dummy observation, $\omega^{\emptyset} \in \Omega^{\mathbf{H}}$, that makes all $\mathbf{R}$'s $\langle a, s \rangle$ pairs seem indistinguishable, where $a \in \mathcal{A}^{\mathbf{R}}, s \in \mathcal{S}$. At any time step, $t \in \{1, \ldots, \mathcal{T}\}$, throughout the joint execution, $\mathcal{B}_t$, which is a set of states represents $\mathbf{H}$'s belief. Here, $\mathcal{T}$ is the last time step of the joint execution. If $|\mathcal{B}_t| = 1$, then $\mathbf{H}$ has full observability of the state at time step t. The belief update is defined as follows: **(1)** at time step $t = 0$, $\mathcal{B}_0 = \{s \mid \exists s \in \mathcal{S}, s = \mathcal{I}\}$, **(2)** at time step $t \in \{1, \ldots, \mathcal{T}\}$, let $\omega_t^{\mathbf{H}} \in \Omega^{\mathbf{H}} \setminus \omega^{\emptyset}$ be the observation received by $\mathbf{H}$, then $\mathcal{B}_t = \{s' \mid \exists s \in \mathcal{B}_{t-1}, a \in \mathcal{A}^{\mathbf{R}} \cup \mathcal{A}^{\mathbf{H}} : \Gamma(a, s) \models s' \wedge \mathcal{O}^{\mathbf{H}}(a, s') = \omega_t^{\mathbf{H}}\}$. If $\omega_t^{\mathbf{H}} = \omega^{\emptyset}$ then $\mathcal{B}_t = \mathcal{B}_{t-1}$. This is because practically the null observation does not reveal any new information in $\mathbf{H}$'s belief update. $\mathbf{H}$ starts its execution from an intermediate belief state. Let $t = k$ be an intermediate time step and $\mathcal{B}_k$ be $\mathbf{H}$'s starting belief for its execution. $\mathbf{R}$ maintains $\mathbf{H}$'s belief state from initial belief until $t = k$.

8.2.2 Formal Guidelines for a Proactive Assistant

An implicit objective of $\mathbf{R}$ is to ensure that $\mathbf{H}$'s cost of achieving its goal is less than that of achieving its goal by itself. We can formalize this intuition about loss/gain in terms of cost experienced by $\mathbf{H}$ when it participates in a joint execution

by using the notion of cost differential. Given a joint plan, $\pi_{\text{MA-COPP}}$, that solves a planning problem for the goal, $G^{\mathbf{H}}$, let $\mathcal{C}^{\mathbf{H}}_{\Delta}(\pi_{\text{MA-COPP}})$ represent the cost differential between the cost incurred by $\mathbf{H}$ when $\pi_{\text{MA-COPP}}$ is executed, versus the minimum cost it incurs when it achieves the goal by itself, i.e., $\mathcal{C}^{\mathbf{H}}_{\Delta}(\pi_{\text{MA-COPP}}) = c^{\mathbf{H}}(\pi_{\text{MA-COPP}}) - c^{\mathbf{H}}(\pi^{*}_{\mathbf{H}})$. For $\mathbf{H}$ to participate in an assistive joint plan with $\mathbf{R}$, it only makes sense if and only if the assistance provides a reduction in her total cost. Otherwise, $\mathbf{H}$ may be better off executing its own plan to its goal. Therefore, for $\mathbf{R}$ to be an assistive agent, the first constraint is to ensure that it only produces a joint plan where the assistance decreases $\mathbf{H}$'s minimum cost (given by $\mathbf{H}$' decision algorithm). That is, for a joint plan $\pi_{\text{MA-COPP}}$, $\mathcal{C}^{\mathbf{H}}_{\Delta}(\pi_{\text{MA-COPP}}) < 0$. In addition, $\mathbf{R}$ should keep track of the belief updates that $\mathbf{H}$ may go through before the start of its execution phase. Given that, $\mathbf{R}$ is aware of $\mathbf{H}$'s sensor model, by simulating the belief it can choose its actions to either limit or increase the amount of information being shared with $\mathbf{H}$. $\mathbf{R}$ can achieve this in multiple ways: (1) by either making certain part of the current state legible (collapsing the states in $\mathbf{H}$'s belief) to reveal particular information to $\mathbf{H}$, or (2) by obfuscating the current state completely thereby keeping some unnecessary complexities hidden from $\mathbf{H}$'s belief. This belief modulation allows $\mathbf{H}$ to participate in the joint plan. As without any awareness about the assistance, $\mathbf{H}$ may tend to follow her original plan. Thus by controlling $\mathbf{H}$'s observability, $\mathbf{R}$ can not only assist $\mathbf{H}$ but also guide it towards a cheaper plan to $G^{\mathbf{H}}$. As a result of this belief modulation, $\mathbf{H}$'s planning problem gets modified to $\mathcal{M}^{\mathbf{H}}_{k} = \langle \mathcal{F}, \mathcal{A}^{\mathbf{H}}, \mathcal{B}_{k}, G^{\mathbf{H}}, c^{\mathbf{H}} \rangle$. Let $\pi_{\mathcal{M}^{\mathbf{H}}_{k}}$ be an minimum cost plan (as per $\mathbf{H}$' decision algorithm) for this modified problem, then $c^{\mathbf{H}}(\pi_{\mathcal{M}^{\mathbf{H}}_{k}}) = \mathcal{C}^{\mathbf{H}}_{\Delta}(\pi_{\text{MA-COPP}})$. Therefore, the second constraint for $\mathbf{R}$ is that, $\mathbf{H}$ should be aware of the reduction in its cost in the modified planning problem.

Finally, the overall effort needed from $\mathbf{H}$'s end to participate in $\pi_{\text{MA-COPP}}$ should be minimized while accounting for both the prior constraints. This is important because,

even though **H** only starts executing after **R**, **H**'s active involvement in the joint plan itself starts from the beginning of the plan. This involves the additional overhead experienced by **H** in updating its belief as a result of **R**'s actions. This penalty incurred by **H** can be formulated in different ways (for e.g., the cost associated with belief update during **R**'s execution, etc). We approximate this penalty as the overall length (time steps) of **R**'s part of the joint execution [2] in addition to **H**'s execution cost. Let $\mathcal{L}$ be the maximum cost that **H** is willing to accommodate in first part of the joint execution, i.e., $k < \mathcal{L}$. Therefore, a proactive assistant optimizes:

$$\min \quad \alpha\ k + (1-\alpha)\ \mathcal{C}_{\Delta}^{\mathbf{H}}(\pi_{\text{MA-COPP}}) \tag{8.1}$$

$$\text{subject to} \quad \mathcal{C}_{\Delta}^{\mathbf{H}}(\pi_{\text{MA-COPP}}) < 0 \tag{8.2}$$

$$c^{\mathbf{H}}(\pi_{\mathcal{M}_k^{\mathbf{H}}}) = \mathcal{C}_{\Delta}^{\mathbf{H}}(\pi_{\text{MA-COPP}}) \tag{8.3}$$

$$k < \mathcal{L} \tag{8.4}$$

where α is a parameter. By setting α appropriately, we can choose joint plans that **H** may prefer in terms effort required.

8.3 Solution Methodology

Although, the overall objective here is to find a joint plan that satisfies equation 8.1, we can only synthesize behavior of the autonomous agent, **R**. We assume that **H** is an independent agent capable of planning towards its own goal. Given that **H** has partial observability of some of the actions performed by **R** and operates in a belief space, we assume that **H** is capable of computing a conformant plan [43; 76] from the belief at the beginning of its execution phase. A conformant plan solves the task by accounting for the relevant uncertainties and does not rely upon being able to get further information from **R**.

[2] Instead of using the entire length of **R**'s part of the joint plan, we can choose to use only the length of observable time steps, i.e. we can choose to ignore the time steps with null observations.

Therefore, a solution to equation 8.1 involves finding a plan for **R** from the initial state to a desirable belief state, $\mathcal{B}_k$ considering the best response of **H** at time step k. Practically, this is a nested search process, where in the outer search loop, the algorithm searches for a desirable belief state by performing a sequence of actions consistent with **R**'s action model. While in the inner search loop (marking the start of phase 2), the algorithm searches for satisfaction of the goal by performing a sequence of actions consistent with **H**'s action model. However, since **H** operates in a belief space, for each node we need to maintain **H**'s belief consistent with that node. And this nested search is essentially a search over a belief space that not only achieves the goal, but also reaches an intermediate partially legible or obfuscatory belief. Since it is not known beforehand, what a desirable belief state would look like for a given problem, it is not that straightforward to design a goal-directed heuristic function to expand the search space. Instead, we use Monte Carlo tree search (MCTS) as a possible way of quickly sampling states and building a utility based tree by performing simulations using a conformant planner for the inner search loop. Once we have access to such a tree, we can then perform search on it by expanding only the high utility search nodes in the tree.

In our approach, we only synthesize for a single agent in a serialized manner. Therefore, there is no need to wait for the moves of the second actor and we can use a single-player version of MCTS [84]. By running numerous quick simulations on the solution space, we can build a sufficiently good utility tree starting from initial state of **R**. The single player MCTS approach for constructing the utility tree is outlined in Algorithm 8. For n iterations, the selection of nodes to be expanded in the tree is done using UCT (upper confidence bound 1 applied to trees) given by $\frac{node-utility}{node-visits+\epsilon} + C * \sqrt{\frac{\ln elapsed-iterations}{node-visits}}$ [55]. The depth of the tree is expanded until **R**'s budget $\mathcal{L}$ (from Equation 8.4) runs out. For each of the expanded nodes, we

Algorithm 8 Generation of utility tree

1: **Input:** MA-COPP, $c^{\mathbf{H}}(\pi^*_{\mathbf{H}})$, $\mathcal{L}$, m (number of iterations)
2: **Output:** *tree* (utility tree)
3: $tree \leftarrow node(\mathcal{I}, \mathcal{B}_0, utility = 0)$
4: **for** m iterations **do**
5: /* select a leaf node using UCT to evaluate nodes */
6: $node, t_{node} = \texttt{select}(tree)$
7: /* expand a child node */
8: $child, t_{child} = \texttt{expand}(node, t_{node})$
9: **if** $t_{child} < \mathcal{L}$ **then**
10: /* create conformant planning problem */
11: $\pi_{\mathbf{H}} = \texttt{planner}(child.\mathcal{B}_t)$
12: /* simulate using conformant plan */
13: **if** $\pi_{\mathbf{H}} \neq \emptyset$ & $c^{\mathbf{H}}(\pi_{\mathbf{H}}) < c^{\mathbf{H}}(\pi^*_{\mathbf{H}})$ & $b_{\mathcal{T}}^{\pi_{\mathbf{H}}} \models G^{\mathbf{H}}$ **then**
14: $reward = \beta$
15: $cost = \alpha \;\; t_{child} + (1 - \alpha)\; c^{\mathbf{H}}(\pi_{\mathbf{H}})$
16: **else**
17: $reward = 0$
18: $cost = \phi$
19: **end if**
20: /* Backpropagate reward and cost */
21: $\texttt{backpropagate}(child, reward - cost * \epsilon)$
22: **end if**
23: **end for**

simulate using a conformant plan generated from node's belief to solve $G^{\mathbf{H}}$. The satisfaction of the goal and the length of the plan, determines the overall reward to

be backpropagated.

The utility tree thus constructed is then used to compute the actual joint plan. In this utility tree, we can consider n best children for each node (i.e. nodes with higher utility and/or higher number of visits). This helps in reducing the solution space with paths that have now been sampled to ensure the satisfaction of all the constraints listed out in equations 8.2 through 8.4. On this reduced search space, we can now perform a simple search to find a node that minimizes the equation 8.1 as well as satisfies the goal. The path to the best such node is then the part of the joint plan that is executed by **R**. This secondary search on the utility tree is only to ensure that the solution minimizes the equation 8.1. Additionally, n can be increased to ensure completeness. Depending on the number of iterations m of MCTS, the value of n can be modulated.

8.4 Evaluation

We conducted a user study to validate the underlying hypothesis of our framework, that the human only recognizes the reduction in her own cost to the goal when the agent takes into account the human's awareness of the assistance. For the user study, we use urban search and rescue (USAR) domain presented in the running example. We also perform an empirical evaluation to analyze the performance of our approach using `USAR` domain and modified IPC `Driverlog` domain.

Domain Setup

Both the `Driverlog` domain and `USAR` domains were written in PDDL (Planning Domain Definition Language) [71]. For both `Driverlog` and `USAR`, we create two versions of the domain: for **R** and **H** respectively. **R**'s version consists of actions that are partially observable as well as non-observable to **H**. Further, for each action, there

are two action definitions in the domain: one to capture **R**'s state transition with full observability as well as the other annotated with keyword "belief" to perform the corresponding belief update for **H**. A parser is used to apply either the belief version of the action (for actions without full observability to **H**) or the regular version of the action (for actions with full observability to **H**). In the "belief" version of the actions, to represent uncertainty over some fluents, we use the standard semantics used in conformant planning benchmarks like "unknown", "oneof" clause to mark a fluent as uncertain. For **H**'s version of the domain, some action definitions that depend on uncertain fluents have conditional effects, written using the standard "when" clause consisting of the condition followed by the effect.

In the `Driverlog` domain, each domain version is associated with a different driver representing **R** and **H**. In this domain, the goal of the drivers is to deliver packages from one city to another. The set of actions available to **R** consist of *stamping-packages-same, stamping-packages-different, load-truck-same-package, load-truck-different-package, unload-truck-same-package, unload-truck-different-package, board-truck, disembark-truck, drive-truck, walk.* Here the first two actions are directly modulating **H**'s observability of packages. If **R** stamps the packages the same, **H** cannot tell the difference between them and has to consider all the packages. While if they are stamped differently, **H** can identify the individual packages. Here depending on the goals of the two agents and on the amount of information **R** wants to convey, it will accordingly choose the stamping actions. The first 6 actions listed above have a "belief" version of the action definition to model **H**'s belief update corresponding to those actions. In **H**'s version of the domain, except the stamping actions, all the other actions listed above are available. Out of these the first 4 actions have conditional effects.

In the `USAR` domain, each domain version is associated with a different agent repre-

senting **R** and **H**. In this domain, the agents' objective is to collect items like medkit, fire extinguisher, etc. from various rooms on the floor. The set of actions available to **R** include *room-cleared, display-wagon-item, move-to-room-wagon-displayed-singleroom, move-to-room-wagon-displayed-connectedroom, move-to-room, move-to-room-wagon, move-to-room-wagon-empty, add-item-wagon, remove-item-wagon, carry-wagon, leave-wagon.* Here, the first two actions modulate which rooms **H** will visit. If clear room action is performed then **H** will not visit those rooms since those rooms have been declared empty. If display action is performed then **H** will visit the single/connected rooms that **R** visits after displaying the wagon items. The next action – moving displayed wagon to single room – has full observability, whereas the next action associated with moving the wagon to a connected room has partial observability, i.e., the wagon can be any of the connected rooms. The rest of the actions have a full observable versus completely non-observable versions – when the human and the agent are in the same room, all of these actions become fully observable otherwise they are not observable. The first four actions have "belief" version of the action definition to model **H**'s belief update corresponding to those actions. In **H**'s version of the domain, there are no wagon related actions, the human can move between the rooms and can pick up and drop from a room and from a wagon. Out of these, the picking up and drop actions have conditional effects.

8.4.1 Empirical Evaluation

We use the approach discussed in Section 3 to generate solutions. We use Conformant-FF planner [3] [42] to simulate **H**'s plans given a belief state. For both the domains, we kept $\mathcal{L} = 15$, i.e., maximum length of **R**'s part of the joint plan. We ran our experiments on 3.5 GHz Intel Core i7 processor with 16 GB RAM. In Table

[3]Source code for Conformant-FF: `https://fai.cs.uni-saarland.de/hoffmann/cff.html`

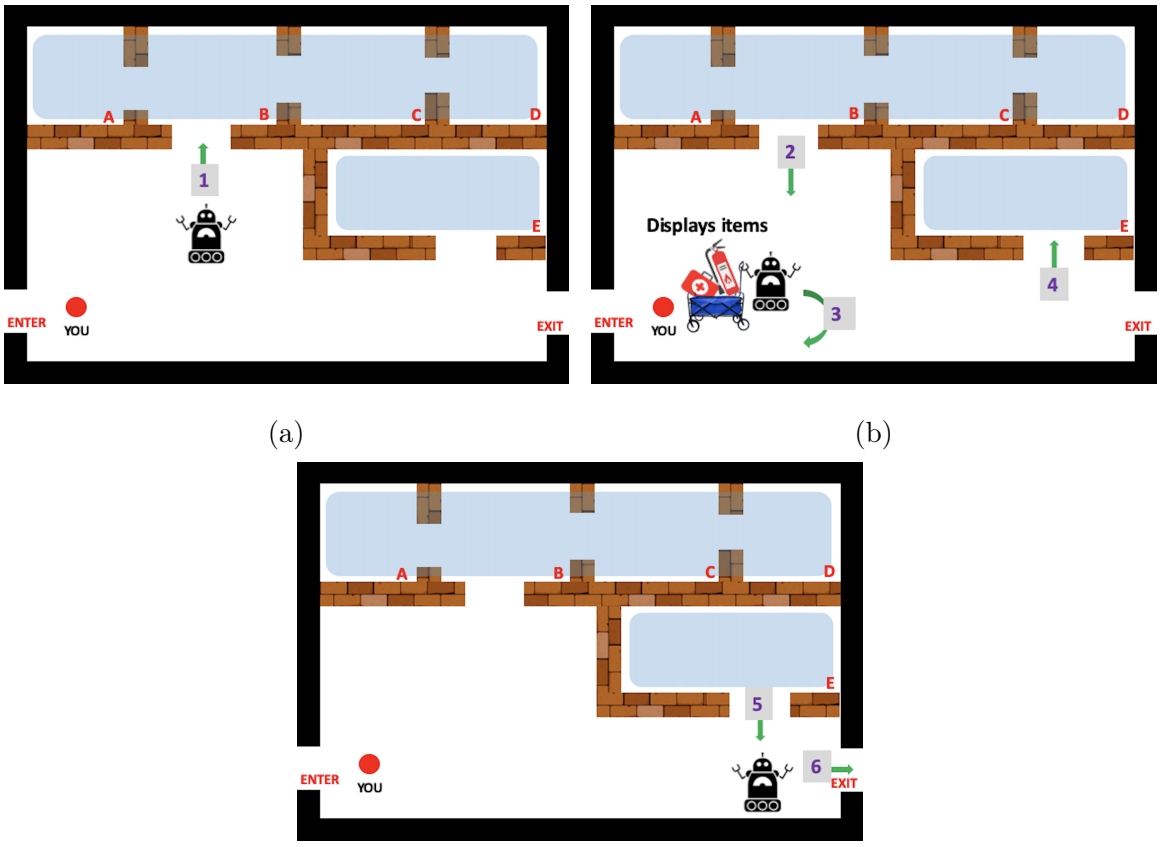
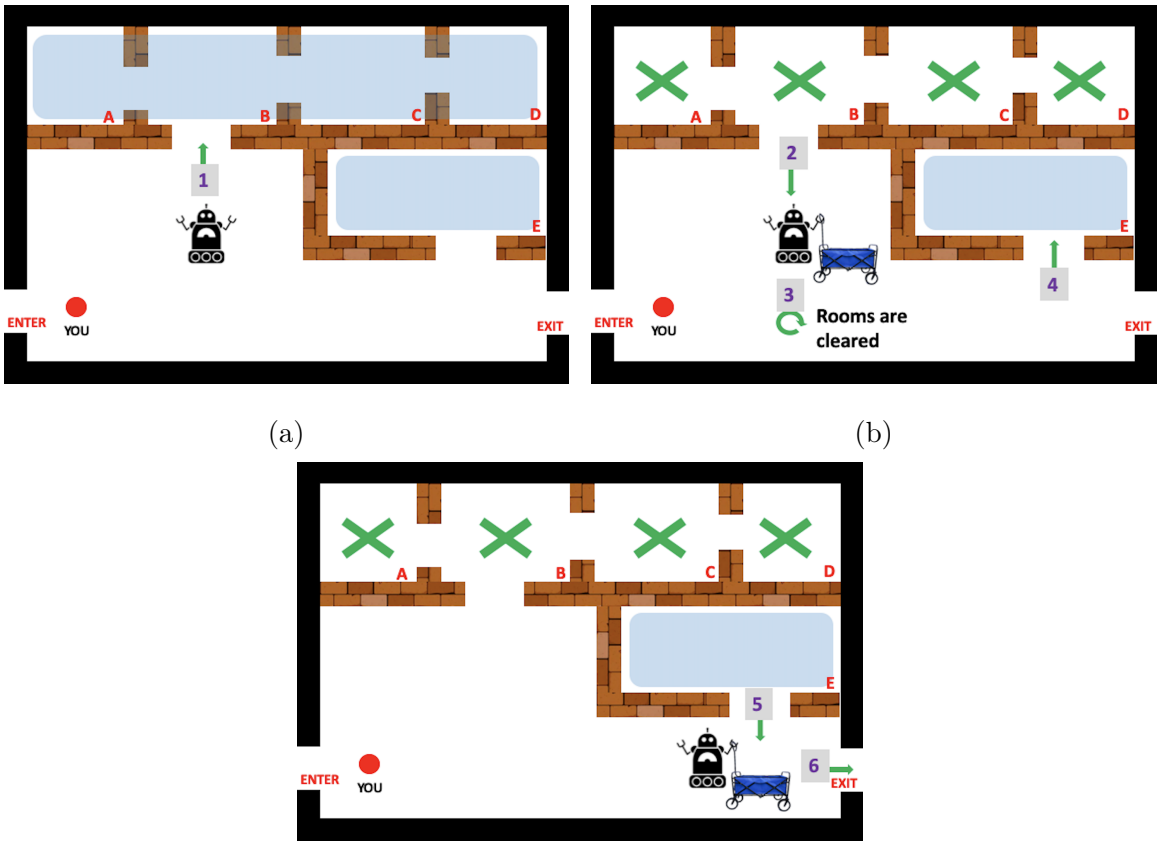


Figure 8.2: Illustration of Assistive Plan Used in First User Study. The Goal of the Human Commander Is to Find a Medkit. She Does Not Know What Items Are Present in Each Room (Indicated by Blue Regions) (a) the Robot Goes into Room B, (b) Comes out with a Wagon and Shows Her the Items of the Wagon. It Then Proceeds to Room E, (c) Comes out Without the Wagon and Exits the Floor.



(a) (b)

(c)

Figure 8.3: Illustration of Assistive Plan Used in the Second User Study. Human's Goal Is to Find All the Medkits. She Does Not Know What Items Are Present in Each Room (Indicated by Blue Regions) (a) the Robot Goes into Room B, (b) Comes out with a Wagon and Declares All Rooms a, B, C, D Are Empty. It Then Proceeds to Room E, (c) Comes out and Exits the Floor with the Wagon.

Domain	#	$c^{\mathbf{H}}(\pi^*_{\mathbf{H}})$	$\alpha = 0.2$			$\alpha = 0.8$			m	Time (sec)
			$c^{\mathbf{H}}(\pi_{\text{MA-COPP}})$	% decrease	$\lvert\pi_{\text{MA-COPP}}\rvert$	$c^{\mathbf{H}}(\pi_{\text{MA-COPP}})$	% decrease	$\lvert\pi_{\text{MA-COPP}}\rvert$		
Driverlog	1	7	3	57.14	8	4	42.85	6	7000	155
	2	8	5	37.5	11	5	37.5	7	7000	171
	3	9	3	66.67	9	4	55.55	7	7000	165
USAR	1	15	3	80	14	5	66.67	8	11000	257
	2	15	4	73.33	12	7	53.33	10	11000	240
	3	12	4	66.67	13	4	66.67	7	11000	254

Table 8.1: Empirical Evaluation Results for Two Domains with for Different α Values (Shows Human Prioritizing Between Processing Load Vs Task Load).

8.1, we report for each problem **H**'s optimal cost without any assistance from **R**, **H**'s cost from participation in joint plan, percentage decrease in **H**'s cost, length of the joint plan, number of iterations used to construct the utility tree and the time taken to generate the solutions. By setting α parameter, we can see how the joint plans prioritize task load vs processing load. We varied n best children from 1 to 5 during the search but the solutions were not impacted thus indicating that the optimal solutions had been found for those problems for $n = 1$. As shown in the table, a steep percentage decrease is obtained for both the domains (specifically for USAR). Additionally, the joint plan itself is not too long even when **R** is assisting. This is because **R** has more information and is capable of guiding **H** in a way that reduces **H**'s cost.

8.4.2 User Study

We conducted two user studies each with a within-subject design to validate the underlying hypothesis. The participants for the studies were recruited from Amazon Mechanical Turk [24]. For each study, we collected 34 submissions. For the first user

study after filtering, we had 31 submissions, and for the second we had 27 submissions. Each participant was paid at the rate of $15/hour for 10 minutes.

Hypothesis 1a *Without legible (revealing information) actions,* **H** *is not aware of the assistance provided by* **R**.

Hypothesis 1b *Both legible (revealing information) and obfuscating (hiding information) actions allow* **H** *to experience reduction in task load and processing load.*

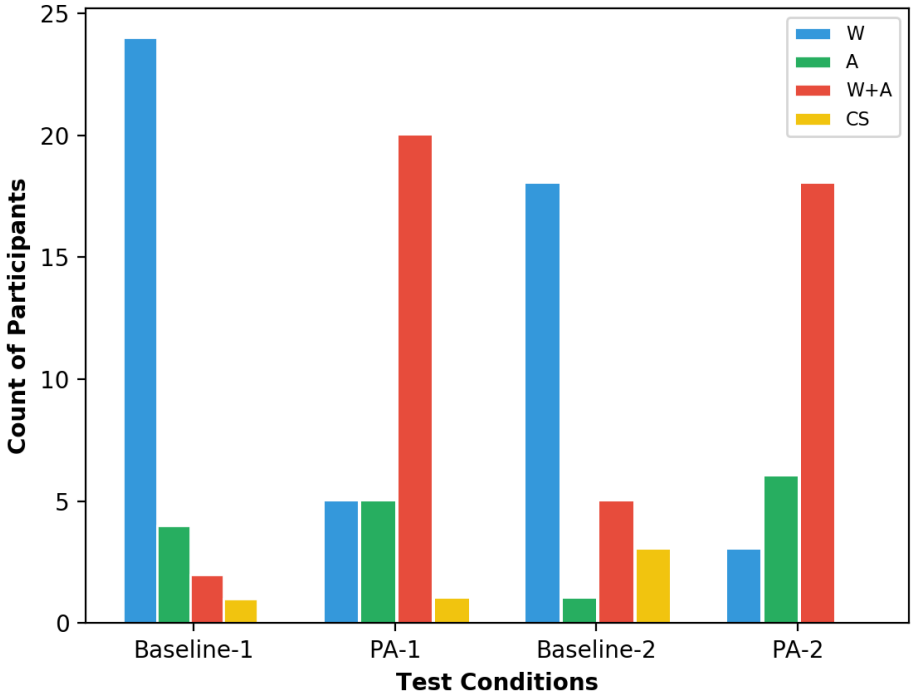


Figure 8.4: Results for Hypothesis 1a. The Four Colors Stand for 4 Options in Questions (3) and (4). Here PA Refers to Proactive Assistant, and 1 and 2 Denote the User Study Numbers.

The format of our studies was as follows: the subjects read through the rules of the `USAR` domain. Then they were shown two scenarios one after another illustrating the robot's behavior. After seeing each scenario they were asked how they would

solve their task. This was also the filter question to make sure they understood the scenario. The submissions which passed the filtering were used to calculate the results. The two scenarios were different in only one action. One scenario satisfied Equation 8.3, the other did not. For USAR, the scenarios that satisfy Equation 8.3 have been discussed in Table 8.1. The order of the scenarios was flipped for half of the participants to account for sequential bias. In user study 1, Figure 8.2 with and without the display action (action number 3 in Figure 8.2b) was shown, while in user study 2, they were shown the illustration in Figure 8.3 with and without the explanation that the rooms are cleared (action number 3 in Figure 8.3b). After the filter question, they were asked to answer a survey: **(1)** rate the scenarios in terms of workload **(2)** rate the scenarios in terms of the effort needed to come up with a plan **(3)** what they thought the robot was doing scenario 1 **(4)** same for scenario 2. For questions (1) and (2), they had to rate the scenarios on a 7 point Likert scale "1 – very hard" to "7 – very easy". For questions (3) and (4), they were given the following 4 options - **(a)** working on its task **(b)** assisting them **(c)** working on its task and assisting them **(d)** cannot say.

For the first user study, we used the illustration similar to Figure 8.1. This is the same problem as the one explained in the running example, except the participants were not aware of the actions performed by the robot within the rooms as shown in Figure 8.2. In this scenario, the robot's behavior hides unnecessary details like initial location of the kit from the human, but important details like final location of medkit are revealed. While for the second user study, we used the illustration in Figure 8.3. Here the human's goal is to find *all* the medkits on the floor. In this case, the robot while picking up items necessary for its goal, also picks up the medkits in rooms A and B (recall that the robot knows the item locations). Further, it reveals to the human that rooms A to D are empty. The robot then drops one medkit in room E

and takes the other medkit by itself. Thereby, hiding complexities like existence of multiple medkits, their initial locations, while revealing information that rooms A to D are empty allowing the human to deduce that the medkits (if any) would be in room E.

In hypothesis 1a, our aim is to check whether the legible actions allow the robot to ensure human's awareness of the assistance. From results of questions (3) and (4) shown in Figure 8.4, we can see that for the baseline behaviors, only 6 (combining both options (b) and (c) referring to assistive behavior) out of 31 participants and 6 out of 27 participants attributed assistive behavior to the robot in study 1 and 2 respectively. In contrast, for behaviors with one extra legible action, 25 out of 31 participants and 24 out of 27 participants attributed assistive behavior to the robot in study 1 and 2 respectively. Since the only difference between the two scenarios is a single legible action, the results confirm our hypothesis that legible actions make **H** aware of **R**'s assistance.

In hypothesis 1b, our aim is to check whether the assistance provided by **R** allows **H** to experience potential reduction in task load and the overhead of processing robot's behavior and coming with a plan to solve the task. For first study, the average score for workload for the baseline behavior was 3.22 (recall that 1 denotes "very hard") in contrast to that of 5.96 for PA (proactive assistant), with a statistical significance (p-value = 0.0000001, p-value < 0.05) obtained by running a two tailed paired t-test, an effect size of 1.89 by running Cohen's d test. While the average score for processing robot's behavior was 3.45 for baseline and 6.06 for PA, with a p-value < 0.05 and effect size of 1.62. For second study, the average score for workload was 2.74 for baseline in contrast to that of 5.55 for PA, with a p-value < 0.05 and effect size of 1.95. The average score for processing robot's behavior was 3.55 for baseline in contrast to 5.85 for PA, with a p-value < 0.05 and effect size of 1.67. All effect

sizes suggest each of the two conditions differ by a large standard deviation. This confirms our hypothesis that the legible and obfuscating actions reduce the overall task and processing load by hiding unnecessary complexities and revealing necessary information.

8.5 Concluding Remarks

In this chapter, we discussed the behavior synthesis of a proactive assistant. While providing proactive assistance, the robot has to ensure that the human is aware of how the assistance affects her task, since the human may not be expecting to receive the assistance in the first place. We presented the multi-agent controlled observability planning (MA-COPP) framework to formulate this problem of proactive assistance. The robot modulates the human's belief to convey the potential reduction in human's cost. We presented a Monte Carlo tree search based solution approach to synthesize this behavior, which samples partial joint plans to construct a utility tree that can be used to optimize the robot's objective of proactive assistance. We validated the underlying hypotheses through user studies and performed empirical evaluation to analyze the performance of our approach.

Chapter 9

CONCLUSION

In the chapters so far, we have seen how the robot can leverage its understanding of the human's mental model to exhibit suitable behaviors either to conform to the human's model, or to communicate information to the human, or to hide information from an adversary. In this chapter, we will conclude our discussion on the synthesis of various interpretable as well as obfuscatory behaviors available to a robot. First, we will summarize the problems this thesis addresses and the approaches presented to solve them. Second, we will reflect on some aspects of the approaches presented as well as point out some avenues for future work. Third, we will highlight key takeaways from this thesis.

9.1 Summary

In this thesis, we have seen that the observer's mental model of the robot model may be different from the robot's actual model either in terms of robot's task, its actions or observability of its actions. And these differences result in confusion over the robot's behavior, or uncertainty about its objectives and intentions. The robot can choose to avoid the confusion as well as limit the uncertainty for its teammates or it can choose to further propel the uncertainty for its enemies. Therefore broadly its behavior can be classified into interpretable and obfuscatory behaviors as seen throughout the previous chapters. We will now summarize each of these behaviors and the approaches used to synthesize them.

In Chapter 3, we discussed the synthesis of explicable plans. An explicable plan allows the robot to minimize the distance between its plan and the human's expecta-

tions of its plan. We discussed two approaches to generate explicable plans: model-based and model-free. The model-based approach uses the human mental model as an input. In this approach, a mapping between the expected plans (i.e., the optimal plans in the human's mental model) and the robot plans is learned using a regression function. To learn such a mapping, explicability scores of the candidate robot plans are collected. Then a set of plan distances between a candidate robot plan and an expected plan are mapped to the explicability score of the candidate plan. Once the regression function is learned, given a distance between a robot plan and an expected plan, the function outputs its explicability score. This regression function is used as a heuristic to guide the explicable plan generation process. In the model-free approach, an explicit human mental model is not required. Instead a conditional random field (CRF) based model is used to capture the underlying patterns which characterize the explicable behaviors for that domain. This model is trained using candidate robot plans labeled by users to indicate explicable and inexplicable parts. Once this model is learned, it can be used to label a given sequence indicating its explicable and non-explicable parts. This learned model is then used as a heuristic to guide the explicable plan generation process. We demonstrated the effectiveness of these approaches in both simulated domains as well as physical robot domains by performing user studies.

In Chapter 4, we discussed legible behaviors that allow the robot to communicate information about its goals (i.e., goal-legible behaviors) and plans (i.e., plan legible behaviors), when the human has partial observability of the robot's activities. By taking the observer's sensor model into account the robot can communicate the necessary information. Here we saw the controlled observability planning (COPP) framework that models the underlying general problem setting. We discussed a general algorithm template that can be used to solve a COPP problem variant. We also discussed the connection between plan legible behaviors and predictable behaviors. We performed

an empirical analysis using IPC domains to evaluate the performance of the two COPP variants as well as to understand the impact of an algorithm parameter on the solution coverage for goal legibility.

In Chapter 5, we discussed an environment design approach that facilitates explicable behavior in environments, which may not be well-suited for it. The environment design for explicability framework selects environment modifications that are optimized for explicable robot behaviors for a given set of tasks that are performed by the robot over a time horizon. In this setting, there is a longitudinal impact on the explicable behaviors since the robot may perform a task repeatedly over a time horizon. We capture it using a discounted Markov reward process. We solve the problem of design for explicability by performing a meta-search in the space of environment configurations, and provide a compilation to classical planning for node evaluation given a cost-based inexplicability score. The compilation solves the problem of generating the most explicable plan in a given environment. We perform empirical analysis over three IPC domains to show (1) improvement of explicability score resulting from environment redesign on problem instances, (2) impact of the design objective parameters on the number of design modifications chosen in the solutions. We also briefly discussed the problem of environment design for communicative behaviors like legibility and predictability and saw that the goal recognition design as well as plan recognition design fall out as special cases of the design problems for legibility and predictability.

In Chapter 6, we discussed different types of obfuscatory behaviors that allow the robot to hide information about its goals (i.e. goal obfuscation) and plans (i.e. plan obfuscation). We discussed an approach to synthesize secure goal obfuscation which maintains obfuscation even when the algorithm is queried with different goal inputs. All of these obfuscatory behaviors are also COPP problem variants. We

evaluated the performance of different goal obfuscation approaches as well as that of different COPP problem variants, and we also analyzed the impact of an algorithm parameter on the solution coverage for goal obfuscation. We also saw the problem of resource bounded goal obfuscation, which allows the robot to ensure that the goal obfuscation is secure and is not biased by any of the decoy goals. We discussed a solution approach to solve this problem which involved choosing appropriate list of decoy goals that have higher similarity to the robot's true goal. Then we used these goals to generate a non-biased secure goal obfuscatory plan by computing equidistant states and bounded-length belief plans to these states. We also performed empirical evaluations using IPC domains to analyze the performance of this approach.

In Chapter 7, we discussed the MO-COPP formulation which is a more general version of the controlled observability planning framework since it supports both adversarial as well as cooperative observers simultaneously. We saw two solution approaches to solve MO-COPP: (1) we formulated the problem as a constraint optimization problem and showed that the MO-COPP can be solved optimally given the time horizon, (2) we showed that it is possible to leverage COPP framework that tackles obfuscation and legibility in isolation to compute satisficing solutions for MO-COPP. We evaluated both of the approaches using 6 domains in total to show the feasibility and utility of the solution approaches.

In Chapter 8, we discussed the problem of proactive assistance, that is a setting where the robot is capable of proactively assisting a human on her task. We showed that the robot can reason over the human's awareness of the assistance by modulating her belief states to reveal necessary information and hide irrelevant information about her goal. Specifically, we presented a set of guidelines that allow the robot to play the role of a *proactive assistant*: **(1)** its activity decreases the human's cost towards her own goal **(2)** the human is able to recognize the potential reduction in her cost

(3) its activity optimizes human's overall cost towards her goal. We then discussed a solution approach for quickly sampling partial joint plans and constructing a utility tree to synthesize desired assistive behaviors. Through user studies and empirical evaluations we validated the underlying hypotheses and analyzed the performance of the algorithm.

9.2 Discussion and Future Work

We will now look at some of the relevant works that were not fully captured in this thesis. We will also remark on several aspects of the presented work that open up avenues for future directions.

9.2.1 Landscape of Robot Behaviors

In a recent survey paper [14] on various interpretable and obfuscatory behaviors, we presented a coherent taxonomy for different robot behaviors. This work was not fully described in this thesis, however it is quite central to the theme of this thesis. The recent research in the area of human-aware AI has typically lacked coherence on the terminologies used to discuss different types of behaviors. A quick scan of the existing literature reveals algorithms for "explicable", "legible", "predictable" and "transparent" planning with overlapping, and sometimes conflicting semantics. The same can be said of a parallel thread of work on the "deception", "privacy" and "security" of plans. This work attempts to provide some clarity and guidance to future researchers looking to work on the topic. In this work, we compared and contrasted existing literature and provided a unified framework for precise specification of these (often confused) ideas. We categorized the interpretable behaviors explored by the research community into three main categories namely: explicability, legibility (or transparency) and predictability. Similarly, we also categorized obfuscatory behav-

iors into four main categories namely: goal obfuscation (or disssimulation, or privacy), deception (or simulation), plan obfuscation, and security. We also highlighted gaps in existing work and directions for future research.

9.2.2 Legibility via Projection-Aware Planning

In this work [18], we explored the use of hologram projections for effective communication of the robot's objectives and intentions during an online interaction between a human and a robot. We showed that, by projecting its intentions as holograms (e.g. by projecting a pickup symbol on a tool that it might use in future), the robot can reduce ambiguity over the possible plans (i.e. projections improve the plan legibility). Further, unlike in traditional mixed reality projection systems, the human can directly interact with these holograms to make her own intentions known to the robot, (e.g. by gazing at and selecting the desired tool thus forcing the robot to replan). Further, we showed that such considerations are not confined to the plan execution phase alone, but can also guide the plan generation process itself by searching for plans that are easier to communicate: i.e. instead of considering only cost optimal plans, the robot can choose plans which are easier to explicate using intention projection actions. We demonstrated the effectiveness of our approach using a physical robot solving a block stacking domain.

9.2.3 Future Directions for Environment Design For Explicability

The environment design framework for explicability presented in Chapter 5 assumes that the robot is capable of performing explicable behavior. However, we can also consider the problem of environment design for explicability when the robot is rational but not cooperative (i.e. it can only generate cost-optimal plans in the given environment and cannot bear the overhead cost of being explicable). In this case, the

emphasis will be on choosing a set of design modifications which reduce the worst case inexplicability score associated with cost-optimal plans for a task. Similarly, we can also consider the problem of environment design for explicability when the robot can communicate (i.e. it can provide an explanation in conjunction with being able to bear the overhead cost of being explicable). In settings involving different humans, the robot will have to provide the same explanation over and over to make its behavior explicable. Therefore, we again see similar trade-offs between one-time design cost versus the cost of repeated explanations borne by the robot. This would require modeling the impact of longitudinal interactions on explanations to account for how the human will update their mental model each time they receive an explanation.

9.2.4 Generalizing MO-COPP *Framework*

The MO-COPP framework presented in Chapter 6 focused on settings with a single adversarial and a single cooperative observer. However, the MO-COPP formulation can be easily generalized to address multiple observers of adversarial type and cooperative type. One way to compute the goal difference, GD, with respect to multiple adversarial and cooperative observers would be to average over the goals achieved for each type of observer and compute the goal difference with respect to the averages. Both the solution approaches are general enough to handle it. In the first approach involving integer programming, the objective function will optimize the average over goals for each type of observer. While in the second approach involving heuristic search, each search node will be required to maintain the beliefs of each corresponding observer.

9.2.5 *Generalizing* MA-COPP *Framework*

The MA-COPP framework described in Chapter 8, Definition 38, can be extended to include the robot's separate planning problem as well (i.e. a separate goal for the robot, say G^R). With this inclusion, the framework becomes quite general and is no longer restricted to assistive planning problem. In fact, it can be used to model cooperative settings like (1) collaborative settings where both the human and robot are working on separate goals that require collaboration as well as non-cooperative settings, like (2) exploitative settings where the robot represents a self-interested agent who wants to reduce its own cost at the expense of the other agent, or (3) adversarial settings where the other agent represents an adversarial entity whose task the robot wants to sabotage. In the collaborative setting, the robot has the objective of minimizing both its optimal cost to its own goal as well as the teammate's optimal cost to her goal by finding a joint collaborative plan that involves guiding the human by modulating her beliefs. In the exploitative setting, the robot's implicit objective is to coerce the other agent to work on the robot's task by withholding some information and deliberately forcing the other agent to reduce robot's cost to the goal. On the other hand, in adversarial setting, the robot's implicit objective is to maximize adversary's cost to goal by modulating adversary's beliefs, even if it comes at an additional cost to the robot. This could be used to model purely adversarial scenarios where the robot may prefer costly punishment [80; 41] over no punishment.

9.2.6 *Assumptions used in the Problem Settings*

We will now briefly discuss some of the implicit assumptions of all the different frameworks discussed in this thesis. Firstly, in all of the discussed frameworks, since the robot's behavior is being observed by the human-in-the-loop, we assume that the

robot is capable of taking into account the human's mental model of itself (which may also involve taking into account human's sensor model). Further, in most of the frameworks (i.e., apart from the MA-COPP framework discussed in Chapter 8), we assume that the human is simply observing the robot with the intent of understanding the robot's behavior, that is, she is observing the robot's behavior without an explicit goal of her own in mind.

In Chapters 3 and 5, we discussed a framework for explicable planning. In this framework, we assume that a human's mental model as well as a distance function is available and/or can be retrieved in some form. However in some of the domains, only partial mental models may be retrievable. Further, different humans in the environment may have competing expectations leading to inconsistent mental models. In these cases, explicable behavior may not be feasible. However explicit communication (in terms of explanations, signaling through projections, interactions in the form question-answers) can be used by the robot to reconcile incomplete models or competing differences in mental models.

In Chapters 4, 6, 7 and 8, we discussed different versions of the controlled observability planning framework. In all of these frameworks, we assumed that the robot has access to the observer's sensor model, and that the observer is capable of updating her beliefs accordingly. If the observer has limited computational capability which prohibits her from updating her beliefs accurately, then the robot will have to account for such computational limitations as well (for instance, allowing belief states of limited size to make it easier for the human to update her belief). Also, we assume a simpler form of observer sensor model that can map the robot's activities to deterministic observations. But there may be settings that require modeling of non-deterministic or stochastic sensor models as well. Further, in Chapter 7, unless the cooperative and adversarial observers have different types of sensor models (say,

due to prior communication or due to inherent differences in observation modalities), it is not possible to simultaneously obfuscate and be legible to different observers. However, in the case where the observers receive the same type observations, if the robot has the capability to explicitly modify the environment in a way that affects their observability, then it can leverage environment design to simultaneously obfuscate and act legibly. Additionally, in Chapter 8, we assume that the robot is aware of the human's model of her task and also has access to her decision making process. The latter assumption may require building a likelihood-based model that captures the human's most likely behavior given a belief state. Then this model can be used to inform the robot's assistive planning process.

9.3 Takeaways

The key takeaways of this thesis are as follows - (1) whenever the robot is acting in the presence of observers, it should take their mental models (inclusive of sensor models) into account. Based on whether the robot is operating in a cooperative or adversarial environment, it should accordingly exhibit interpretable or obfuscatory behaviors. (2) In cooperative environments, if the robot doesn't want to communicate any information, it should act explicably. That is, it should conform to the human's mental model of itself. (3) In cooperative environments, if the robot wants to communicate information about its model, it should perform legible behaviors (either goal or plan legible). (4) The environment in which the robot and the human are coexisting can be redesigned to facilitate interpretable behaviors - like explicable, legible or predictable behaviors. (5) In cooperative environments, if the robot wants to assist, it should ensure that the human is aware of the potential reduction in her cost. (6) In adversarial environments, if the robot wants to hide information about its model, it should perform obfuscatory behaviors (either goal or plan obfuscatory).

(7) In mixed environments with both cooperative as well as adversarial entities, the robot should balance the amount of information shared with its teammates with the amount of information shared with its adversaries.

REFERENCES

[1] Alkhazraji, Y., M. Frorath, M. Grützner, T. Liebetraut, M. Ortlieb, J. Seipp, T. Springenberg, P. Stahl and J. Wülfing, "Pyperplan", https://bitbucket.org/malte/pyperplan (2016).

[2] Bansal, G., B. Nushi, E. Kamar, W. S. Lasecki, D. S. Weld and E. Horvitz, "Beyond accuracy: The role of mental models in human-ai team performance", in "Proceedings of the AAAI Conference on Human Computation and Crowdsourcing", vol. 7, pp. 2–11 (2019).

[3] Bansal, G., B. Nushi, E. Kamar, D. S. Weld, W. S. Lasecki and E. Horvitz, "Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff", in "Proceedings of the AAAI Conference on Artificial Intelligence", vol. 33, pp. 2429–2437 (2019).

[4] Bartlett, C. E. and N. J. Cooke, "Human-robot teaming in urban search and rescue", in "Proceedings of the Human Factors and Ergonomics Society Annual Meeting", vol. 59, pp. 250–254 (SAGE Publications Sage CA: Los Angeles, CA, 2015).

[5] Benton, J., D. Smith, J. Kaneshige, L. Keely and T. Stucky, "Chap-e: A plan execution assistant for pilots", in "Proceedings of the International Conference on Automated Planning and Scheduling", vol. 28 (2018).

[6] Bonet, B. and H. Geffner, "Belief tracking for planning with sensing: Width, complexity and approximations", Journal of Artificial Intelligence Research **50**, 923–970 (2014).

[7] Bonisoli, A., A. E. Gerevini, A. Saetti and I. Serina, "A privacy-preserving model for the multi-agent propositional planning problem", in "Proceedings of the Twenty-first European Conference on Artificial Intelligence", pp. 973–974 (2014).

[8] Brafman, R. I., "A privacy preserving algorithm for multi-agent planning and search.", in "IJCAI", pp. 1530–1536 (2015).

[9] Bryce, D., "Landmark-based plan distance measures for diverse planning.", in "ICAPS", (2014).

[10] Buckingham, D. and M. Scheutz, "Getting help without asking: Stigmergic planning for human-robot collaboration", in "Proceedings of the AAMAS Workshop of Multi-Agent Interaction without Prior Coordination", (2017), short paper.

[11] Carberry, S., "Techniques for plan recognition", User Modeling and User-Adapted Interaction **11**, 1-2, 31–48 (2001).

[12] Chakraborti, T., G. Briggs, K. Talamadupula, Y. Zhang, M. Scheutz, D. Smith and S. Kambhampati, "Planning for serendipity", in "IROS", (2015).

[13] Chakraborti, T., G. Briggs, K. Talamadupula, Y. Zhang, M. Scheutz, D. Smith and S. Kambhampati, "Planning for serendipity", in "2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 5300–5306 (IEEE, 2015).

[14] Chakraborti, T., A. Kulkarni, S. Sreedharan, D. Smith and S. Kambhampati, "Explicability? Legibility? Predictability? Transparency? Privacy? Security?: The Emerging Landscape of Interpretable Agent Behavior", in "ICAPS", (2019).

[15] Chakraborti, T., A. Kulkarni, S. Sreedharan, D. E. Smith and S. Kambhampati, "Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior", in "Proceedings of the international conference on automated planning and scheduling", vol. 29, pp. 86–96 (2019).

[16] Chakraborti, T., S. Sreedharan, S. Grover and S. Kambhampati, "Plan explanations as model reconciliation – an empirical study", in "2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)", pp. 258–266 (2019).

[17] Chakraborti, T., S. Sreedharan, A. Kulkarni and S. Kambhampati, "Alternative modes of interaction in proximal human-in-the-loop operation of robots", arXiv preprint arXiv:1703.08930 (2017).

[18] Chakraborti, T., S. Sreedharan, A. Kulkarni and S. Kambhampati, "Projection-aware task planning and execution for human-in-the-loop operation of robots in a mixed-reality workspace", in "2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 4476–4482 (IEEE, 2018).

[19] Chakraborti, T., S. Sreedharan, Y. Zhang and S. Kambhampati, "Plan explanations as model reconciliation: Moving beyond explanation as soliloquy", in "IJCAI", (2017).

[20] Chakraborti, T., Y. Zhang, D. E. Smith and S. Kambhampati, "Planning with resource conflicts in human-robot cohabitation", in "Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems", (2016).

[21] Christensen, H. I., T. Batzinger, K. Bekris, K. Bohringer, J. Bordogna, G. Bradski, O. Brock, J. Burnstein, T. Fuhlbrigge, R. Eastman *et al.*, "A Roadmap for US Robotics: From Internet to Robotics", Technical Report (2009).

[22] Cirillo, M., L. Karlsson and A. Saffiotti, "Human-aware task planning: An application to mobile robots", ACM Transactions on Intelligent Systems and Technology (TIST) **1**, 2, 15 (2010).

[23] Cohen, P. R., C. R. Perrault and J. F. Allen, "Beyond question answering", Strategies for natural language processing **245274** (1981).

[24] Crowston, K., "Amazon Mechanical Turk: A Research Tool for Organizations and Information Systems Scholars", in "Shaping the Future of ICT Research. Methods and Approaches", (Springer, 2012).

[25] Csibra, G. and G. Gergely, "'obsessed with goals': Functions and mechanisms of teleological interpretation of actions in humans", Acta psychologica **124**, 1, 60–78 (2007).

[26] Dragan, A., K. Lee and S. Srinivasa, "Legibility and predictability of robot motion", in "Human-Robot Interaction", (2013).

[27] Dragan, A. and S. Srinivasa, "Generating Legible Motion", in "RSS", (2013).

[28] Dragan, A. and S. Srinivasa, "Generating legible motion", in "Proceedings of Robotics: Science and Systems", (Berlin, Germany, 2013).

[29] Dragan, A. D., S. Bauman, J. Forlizzi and S. S. Srinivasa, "Effects of robot motion on human-robot collaboration", in "2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)", pp. 51–58 (IEEE, 2015).

[30] Dragan, A. D., K. C. Lee and S. S. Srinivasa, "Legibility and Predictability of Robot Motion", in "HRI", (2013).

[31] E-Martin, Y., M. D. R-Moreno and D. E. Smith, "A fast goal recognition technique based on interaction estimates", in "Twenty-Fourth International Joint Conference on Artificial Intelligence", (2015).

[32] Fan, X., S. Oh, M. McNeese, J. Yen, H. Cuevas, L. Strater and M. R. Endsley, "The influence of agent reliability on trust in human-agent collaboration", in "ECCE", (2008).

[33] Fern, A., S. Natarajan, K. Judah and P. Tadepalli, "A decision-theoretic model of assistance", Journal of Artificial Intelligence Research **50**, 71–104 (2014).

[34] Fisac, J. F., C. Liu, J. B. Hamrick, S. S. Sastry, J. K. Hedrick, T. L. Griffiths and A. D. Dragan, "Generating Plans that Predict Themselves", in "WAFR", (2018).

[35] Geffner, H. and B. Bonet, "A concise introduction to models and methods for automated planning", Synthesis Lectures on Artificial Intelligence and Machine Learning **8**, 1, 1–141 (2013).

[36] Geffner, H. and N. Lipovetzky, "Width and serialization of classical planning problems", (2012).

[37] Grosz, B. and S. Kraus, "Collaborative plans for complex group action", Artificial Intelligence (1996).

[38] Gurobi Optimization, L., "Gurobi optimizer reference manual", URL `http://www.gurobi.com` (2018).

[39] Haslum, P. and P. Jonsson, "Some results on the complexity of planning with incomplete information", in "European Conference on Planning", pp. 308–318 (Springer, 1999).

[40] Helmert, M., "The fast downward planning system.", J. Artif. Intell. Res.(JAIR) **26**, 191–246 (2006).

[41] Henrich, J., R. McElreath, A. Barr, J. Ensminger, C. Barrett, A. Bolyanatz, J. C. Cardenas, M. Gurven, E. Gwako, N. Henrich *et al.*, "Costly punishment across human societies", Science **312**, 5781, 1767–1770 (2006).

[42] Hoffmann, J. and R. Brafman, "Conformant planning via heuristic forward search: A new approach", **170**, 6–7, 507–541 (2006).

[43] Hoffmann, J. and R. I. Brafman, "Conformant planning via heuristic forward search: A new approach", Artificial Intelligence **170**, 6-7, 507–541 (2006).

[44] Hoffmann, J. and B. Nebel, "The ff planning system: Fast plan generation through heuristic search", J. Artif. Int. Res. **14**, 1, 253–302, URL `http://dl.acm.org/citation.cfm?id=1622394.1622404` (2001).

[45] Horvitz, E. J., A. Jacobs and D. Hovel, "Attention-sensitive alerting", arXiv preprint arXiv:1301.6707 (2013).

[46] Kamar, E., Y. Gal and B. J. Grosz, "Incorporating helpful behavior into collaborative planning", in "Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)", (Springer Verlag, 2009).

[47] Keren, S., A. Gal and E. Karpas, "Goal recognition design.", in "ICAPS", (2014).

[48] Keren, S., A. Gal and E. Karpas, "Goal recognition design for non-optimal agents", in "Proceedings of the AAAI Conference on Artificial Intelligence", vol. 29 (2015).

[49] Keren, S., A. Gal and E. Karpas, "Goal recognition design with non-observable actions.", in "AAAI", pp. 3152–3158 (2016).

[50] Keren, S., A. Gal and E. Karpas, "Privacy preserving plans in partially observable environments.", in "IJCAI", pp. 3170–3176 (2016).

[51] Keren, S., A. Gal and E. Karpas, "Strong stubborn sets for efficient goal recognition design", in "Twenty-Eighth International Conference on Automated Planning and Scheduling", (2018).

[52] Keren, S., L. Pineda, A. Gal, E. Karpas and S. Zilberstein, "Equi-Reward Utility Maximizing Design in Stochastic Environments", in "IJCAI", (2017).

[53] Keyder, E. and H. Geffner, "Heuristics for planning with action costs revisited.", in "ECAI", pp. 588–592 (2008).

[54] Knepper, R. A., C. I. Mavrogiannis, J. Proft and C. Liang, "Implicit communication in a joint action", in "Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction", pp. 283–292 (ACM, 2017).

[55] Kocsis, L. and C. Szepesvári, "Bandit based monte-carlo planning", in "European conference on machine learning", pp. 282–293 (Springer, 2006).

[56] Kube, C. R. and H. Zhang, "Task modelling in collective robotics", Autonomous Robots **4**, 1, 53–72 (1997).

[57] Kulkarni, A., T. Chakraborti, Y. Zha, S. G. Vadlamudi, Y. Zhang and S. Kambhampati, "Explicable Robot Planning as Minimizing Distance from Expected Behavior", in "AAMAS", (2019), extended Abstract.

[58] Kulkarni, A., S. Sreedharan, S. Keren, T. Chakraborti, D. Smith and S. Kambhampati, "Designing environments conducive to interpretable robot behavior", in "2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", (IEEE, 2020).

[59] Kulkarni, A., S. Srivastava and S. Kambhampati, "A unified framework for planning in adversarial and cooperative environments", in "AAAI", (2019).

[60] Kulkarni, A., S. Srivastava and S. Kambhampati, "Signaling friends and head-faking enemies simultaneously: Balancing goal obfuscation and goal legibility", in "Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems", AAMAS '20, p. 1889–1891 (International Foundation for Autonomous Agents and Multiagent Systems, 2020).

[61] Kyle Hollins Wray, S. Z., Stefan J. Witwicki, "Online decision-making for scalable autonomous systems", in "Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17", pp. 4768–4774 (2017), URL `https://doi.org/10.24963/ijcai.2017/664`.

[62] Lafferty, J., A. McCallum and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data", (2001).

[63] Langley, P., "Explainable agency in human-robot interaction", in "AAAI Fall Symposium Series", (2016).

[64] Lindell, Y., "Secure multiparty computation for privacy preserving data mining", in "Encyclopedia of Data Warehousing and Mining", pp. 1005–1009 (IGI Global, 2005).

[65] Luis, N. and D. Borrajo, "Plan merging by reuse for multi-agent planning", Distributed and Multi-Agent Planning p. 38 (2014).

[66] MacNally, A. M., N. Lipovetzky, M. Ramirez and A. R. Pearce, "Action selection for transparent planning", in "Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems", pp. 1327–1335 (International Foundation for Autonomous Agents and Multiagent Systems, 2018).

[67] Mainprice, J., E. A. Sisbot, L. Jaillet, J. Cortés, R. Alami and T. Siméon, "Planning human-aware motions using a sampling-based costmap planner", in "Robotics and Automation (ICRA), 2011 IEEE International Conference on", pp. 5012–5017 (IEEE, 2011).

[68] Maliah, S., G. Shani and R. Stern, "Stronger privacy preserving projections for multi-agent planning.", in "ICAPS", pp. 221–229 (2016).

[69] Masters, P. and S. Sardina, "Cost-based goal recognition for path-planning", in "Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems", pp. 750–758 (2017).

[70] Masters, P. and S. Sardina, "Deceptive path-planning", in "Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17", pp. 4368–4375 (2017), URL `https://doi.org/10.24963/ijcai.2017/610`.

[71] McDermott, D., M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld and D. Wilkins, "Pddl-the planning domain definition language", (1998).

[72] Mirsky, R., K. Gal, R. Stern and M. Kalech, "Goal and plan recognition design for plan libraries", TIST (2019).

[73] Narahari, Y., *Game Theory and Mechanism Design*, vol. 4 (World Scientific, 2014).

[74] Nguyen, T. A., M. Do, A. E. Gerevini, I. Serina, B. Srivastava and S. Kambhampati, "Generating diverse plans to handle unknown and partially known user preferences", Artificial Intelligence **190**, 0, 1 – 31 (2012).

[75] Oh, J., F. Meneguzzi, K. P. Sycara and T. Norman, "Antipa: an agent architecture for intelligent information assistance.", in "ECAI", pp. 1055–1056 (2010).

[76] Palacios, H. and H. Geffner, "Compiling uncertainty away in conformant planning problems with bounded width", Journal of Artificial Intelligence Research **35**, 623–675 (2009).

[77] Pereira, R. F., N. Oren and F. Meneguzzi, "Landmark-based heuristics for goal recognition", in "Thirty-First AAAI Conference on Artificial Intelligence", (2017).

[78] Ramırez, M. and H. Geffner, "Plan recognition as planning", in "Proceedings of the 21st international joint conference on Artifical intelligence. Morgan Kaufmann Publishers Inc", pp. 1778–1783 (2009).

[79] Ramırez, M. and H. Geffner, "Probabilistic plan recognition using off-the-shelf classical planners", in "Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI 2010)", (2010).

[80] Rankin, D. J., M. dos Santos and C. Wedekind, "The evolutionary significance of costly punishment is still to be demonstrated", Proceedings of the National Academy of Sciences **106**, 50, E135–E135 (2009).

[81] Rintanen, J., "Complexity of planning with partial observability.", in "ICAPS", pp. 345–354 (2004).

[82] Sadigh, D., S. Sastry, S. A. Seshia and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions.", in "Robotics: Science and Systems", (2016).

[83] Savitch, W. J., "Relationships between nondeterministic and deterministic tape complexities", Journal of computer and system sciences **4**, 2, 177–192 (1970).

[84] Schadd, M. P., M. H. Winands, H. J. Van Den Herik, G. M.-B. Chaslot and J. W. Uiterwijk, "Single-player monte-carlo tree search", in "International Conference on Computers and Games", pp. 1–12 (Springer, 2008).

[85] Sengupta, S., T. Chakraborti, S. Sreedharan, S. G. Vadlamudi and S. Kambhampati, "Radar-a proactive decision support system for human-in-the-loop planning", AAAI Fall Symposium on Human-Agent Groups: Studies, Algorithms and Challenges (2017).

[86] Shekhar, S. and R. I. Brafman, "Representing and planning with interacting actions and privacy", in "Twenty-Eighth International Conference on Automated Planning and Scheduling", (2018).

[87] Sisbot, E. A., L. F. Marin-Urias, R. Alami and T. Simeon, "A human aware mobile robot motion planner", IEEE Transactions on Robotics **23**, 5, 874–883 (2007).

[88] Sohrabi, S., A. V. Riabov and O. Udrea, "Plan recognition as planning revisited.", in "IJCAI", pp. 3258–3264 (2016).

[89] Sreedharan, S., T. Chakraborti and S. Kambhampati, "Balancing explicability and explanation in human-aware planning", in "2017 AAAI Fall Symposium", pp. 61–68 (AI Access Foundation, 2017).

[90] Sreedharan, S., T. Chakraborti, C. Muise and S. Kambhampati, "Planning with explanatory actions: A joint approach to plan explicability and explanations in human-aware planning", in "AAAI", (2020).

[91] Sreedharan, S., A. O. Hernandez, A. P. Mishra and S. Kambhampati, "Model-free model reconciliation", in "Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19", pp. 587–594 (International Joint Conferences on Artificial Intelligence Organization, 2019), URL `https://doi.org/10.24963/ijcai.2019/83`.

[92] Sreedharan, S., S. Kambhampati *et al.*, "Handling model uncertainty and multiplicity in explanations via model reconciliation", in "Proceedings of the International Conference on Automated Planning and Scheduling", vol. 28 (2018).

[93] Srivastava, B., T. A. Nguyen, A. Gerevini, S. Kambhampati, M. B. Do and I. Serina, "Domain independent approaches for finding diverse plans.", in "IJCAI", pp. 2016–2022 (2007).

[94] Sutton, R. S. and A. G. Barto, *Reinforcement learning: An introduction* (MIT press, 2018).

[95] Unhelkar, V. and J. Shah, "Contact: Deciding to communicate during time-critical collaborative tasks in unknown, deterministic domains", in "Proceedings of the AAAI Conference on Artificial Intelligence", vol. 30 (2016).

[96] Vallacher, R. R. and D. M. Wegner, "What do people think they're doing? action identification and human behavior.", Psychological review **94**, 1, 3 (1987).

[97] Štolba, M., *Reveal or Hide: Information Sharing in Multi-Agent Planning*, Ph.D. thesis, Czech Technical University (2017).

[98] Wayllace, C., P. Hou and W. Yeoh, "New Metrics and Algorithms for Stochastic Goal Recognition Design Problems", in "IJCAI", (2017).

[99] Wayllace, C., P. Hou, W. Yeoh and T. C. Son, "Goal Recognition Design with Stochastic Agent Action Outcomes", in "IJCAI", (2016).

[100] Yorke-Smith, N., S. Saadati, K. Myers and D. Morley, "The design of a proactive personal agent for task management", Int. J. Artif. Intell. Tools **21** (2012).

[101] Zhang, H., Y. Chen and D. C. Parkes, "A general approach to environment design with one agent", in "IJCAI", (2009).

[102] Zhang, Y., S. Sreedharan, A. Kulkarni, T. Chakraborti, H. H. Zhuo and S. Kambhampati, "Plan explicability and predictability for robot task planning", in "International Conference on Robotics and Automation (ICRA)", (2017).

[103] Zhuo, H. H. and S. Kambhampati, "Action-model acquisition from noisy plan traces", in "Twenty-Third International Joint Conference on Artificial Intelligence", (2013).

[104] Zhuo, H. H. and Q. Yang, "Action-model acquisition for planning via transfer learning", Artificial intelligence **212**, 80–103 (2014).