ACTIVE GOAL RECOGNITION DESIGN

BY

Kevin C. Gall
BFA in Dramatic Arts, New York University, 2011

THESIS

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Master of Science
in
Computer Science

May, 2021

This thesis has been examined and approved in partial fulfillment of the requirements for the degree of Master of Science in Computer Science by:

Thesis Director, Wheeler Ruml
Professor of Computer Science
University of New Hampshire

Marek Petrik
Assistant Professor of Computer Science
University of New Hampshire

Sarah Keren
Post-Doctoral Fellow of Computer Science
Harvard University

On April 23, 2021

Approval signatures are on file with the University of New Hampshire Graduate School.

DEDICATION

For my mother, who taught me the value of education.

For my mother-in-law, without whom I would not have had the time to complete this degree.

For my children, whom I hope will at least learn to code.

But mostly, for my wife, who has put up with far more than her fair share.

ACKNOWLEDGEMENTS

Thanks to Bence Cserna for his open-source planning research codebase, upon which all code and evaluations in this thesis were built.

Thanks to Wheeler Ruml for guiding my research for years and for advising this thesis.

Thanks to Sarah Keren for offering advice and guidance on my research in the context of her broader work on Goal Recognition Design.

And thanks again to Bence Cserna and Wheeler Ruml for recruiting me into UNH AI research. I can honestly say it changed my life for the better.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

α — Sequence of alternating observer and subject actions that describes the evolution of an Active Goal Recognition Design (AGRD) problem instance.

ϕ — A subject plan, i.e. a sequence of actions the subject could execute assuming no observer interventions.

H_s — Goal hypothesis for a state s, i.e. the set of goals the subject could still be pursuing.

P_s — The probability distribution over all subject actions applicable in state s.

ψ — Function $S \to [0, 1]$ that maps states to a real number between 0 and 1. This is the fraction of a subject's plan that has been executed at some point in the progression of an AGRD problem instance.

d — Notation used to represent the concept of "distinctiveness," a measurement used in multi-goal settings of how long a path can be before no other goal is consistent with that path.

ρ — Generic value function $S \to \mathbb{R}$ that maps states to real numbers. Used in defining the quality of an Active Goal Recognition Design solution.

e_ρ — Expectation of ρ value for some state.

T_{e_ρ} — The tree induced by evaluating e_ρ on some state.

κ — The depth of T_{e_ρ}

T^i_{IW} — The tree explored by the *Iterative Widening AGRD* algorithm on the ith iteration of a search episode.

ABSTRACT

Active Goal Recognition Design

by

Kevin C. Gall

University of New Hampshire, May, 2021

In Goal Recognition Design (GRD), the objective is to modify a domain to facilitate early detection of the goal of a subject agent. Most previous work studies this problem in the offline setting, in which the observing agent performs its interventions before the subject begins acting. In this thesis, we generalize GRD to the online setting in which time passes and the observer's actions are interleaved with those of the subject. We illustrate weaknesses of existing metrics for GRD and propose an alternative better suited to online settings. We provide a formal definition of this Active GRD (AGRD) problem and propose both an optimal algorithm and a suboptimal algorithm for solving it. AGRD occupies an interesting middle ground between passive goal recognition and strategic two-player game settings.

CHAPTER 1

Introduction

Goal recognition is the problem of identifying the goal of a subject agent from observations of its actions [Sukthankar *et al.*, 2014]. The inputs to a goal recognition problem are a domain, a set of possible goals, and observations of a subject acting in the domain. The output is the subject's goal or a probability distribution over possible goals. Applications range from educational games [Ha *et al.*, 2011], in which an education system may wish to discover a student's goal to be able to adapt the learning experience dynamically, to adversarial settings [Ang *et al.*, 2017], in which a defense system may try to predict the target of an attacker.

In this thesis, we study Goal Recognition Design (GRD) [Keren *et al.*, 2014], where the objective is to modify a domain to facilitate early detection of the goal of a subject agent acting to achieve one of a known finite set of goals. The inputs to a GRD problem are a domain, a set of possible goals, a set of interventions we are allowed to execute that modify the domain, a budget that constrains the interventions we can execute, and a starting state for an agent acting in this domain. The output is an ordered sequence of interventions to execute on the domain. While most previous work studies this problem in the offline setting, we propose a variant, Active Goal Recognition Design (AGRD), in which the observing agent executes its interventions online while the subject is acting. The thesis we pursue in this research is:

> Active Goal Recognition Design is a useful new problem setting that sits between passive Goal Recognition and two-player games.

In contrast to most work in Goal Recognition and GRD, where the observer is removed from the subject's execution through the system, AGRD models the interaction between the observer and the subject explicitly, drawing on game-theoretic approaches to multi-agent systems [Shoham and Leyton-Brown, 2009]. However, full 2-player games model a host of dynamics under the assumption that the two players act strategically to maximize their own utility. Instead, we are only concerned with neutral observers and non-strategic subjects. These assumptions model domains where the observer is not trying to interfere with the subject, but is able to reason about their interactions to facilitate goal recognition as quickly as possible.

For example, consider a playground with an area that includes a sand pit for toddlers and an area with tall slides that are dangerous for younger children. Now consider an arriving toddler very excited to play and not listening to her caregiver. She might run toward *both* the sand pit and the tall slide, since they are in the same direction. As those with small children know, toddlers can have a strong independent streak. Her caregiver would want to know which was her goal as soon as possible so that they could intervene before she gets to the slide, but take no other action if she is going to the sand pit so as to avoid a tantrum. Conventional offline GRD would require the designers of the playground to plan in advance a wall separating the two areas. In this case, the toddler must decide which direction she is going in plenty of time for the caregiver to see and react. In the absence of such specific foresight by the designer, the caregiver must step in the middle of the path, forcing the child to go left or right and thus revealing her goal. Clearly, real-world environments do not always have the luxury of offline design efforts to optimize the recognition of agents' goals. Further, there is a wide range of applications where online interaction can supplement offline GRD from caregiving to automated remote assistance [Oh *et al.*, 2014], and even cyber security [Boddy *et al.*, 2005; Edelkamp *et al.*, 2009].

The rest of this thesis will be structured as follows:

- In Chapter 2, we describe the prior work this thesis builds on.

- In Chapter 3, we illustrate limitations of the conventional approaches to GRD and discuss the challenges of adapting Goal Recognition Design problems to online settings.

- In Chapter 4, we define a formal problem setting for the Active Goal Recognition Design (AGRD) problem and present theoretical analysis of its characteristics.
- In Chapter 5, we propose an algorithm, *Opt-AGRD*, for solving AGRD problems, prove its optimality, and evaluate its performance on a set of benchmark domains.
- In Chapter 6, we propose *IW-AGRD*, another algorithm based on *Opt-AGRD*, that exhibits anytime behavior and computes suboptimal solutions by exploring a reduced state space. We evaluate its performance relative to *Opt-AGRD*.
- In Chapter 7, we conclude with a discussion of related work and possible directions for further research.

CHAPTER 2

Prior Work

2.1 Plan Recognition

Goal recognition is closely related to plan recognition, the problem of identifying an agent's plan either after or during execution. Many approaches to plan recognition use plan libraries or policies known a priori, to which observations are matched [Bui, 2003; Avrahami-Zilberbrand and Kaminka, 2014; Kabanza *et al.*, 2010; Mirsky *et al.*, 2018]. However, quality plan libraries can be difficult to acquire for new domains. In our work, we do not assume the existence of plan libraries, and so must model possible behaviors another way. Plan Recognition as Planning [Ramírez and Geffner, 2009] is a technique that matches observations to plans generated using classical planners, shifting the burden from acquiring quality plan libraries to crafting planning models robust enough to capture enough details of subject behavior. This work has been applied to probabilistic and stochastic settings [Ramírez and Geffner, 2010; Oh *et al.*, 2014] and lays the foundation for Goal Recognition Design, which builds on the idea of subjects as agents in a planning domain to identify and eliminate plans that make goal recognition difficult. Masters and Sardina [2019] study Goal Recognition, building on the work of Ramírez and Geffner. They compute target areas, based on distance to the goals, where an agent's presence would indicate with high probability that they are pursuing a specific goal.

2.2 Goal Recognition Design

Goal Recognition Design (GRD) was first proposed by Keren *et al.* [2014], then extended for sub-optimal agents, partial observability, and partially-informed agents [Keren *et al.*, 2015; Keren *et al.*, 2016a; Keren *et al.*, 2016b; Keren *et al.*, 2020]. In a GRD problem, the observer executes interventions in the domain, typically by adding or removing actions, that alter the possible plans the subject can execute. The number of interventions an observer can execute is usually constrained by a user-specified budget. The objective is to minimize worst-case distinctiveness (wcd), the length of the longest (or more generally, the cost of the most expensive) possible plan prefix before the subject's true goal is the only possible goal given the prefix. Possible plan prefixes depend on the setting, but in the simplest case are limited to prefixes of optimal plans. A non-distinct plan prefix is a prefix compatible with multiple goals the agent could be pursuing. Solutions to GRD problems are sequences of interventions in the domain that do not exceed the user's budget and that change the set of plans available to agents to minimize wcd.

Wayllace *et al.* [2017] modify GRD to handle stochastic domains (S-GRD) where the subject agent's actions' outcomes are not deterministic. They propose a new objective, expected case distinctiveness (ecd), to deal with stochasticity. ecd is the expected time that an agent's path will be non-distinct given that the agent is following an optimal policy to its goal. By using an expectation, their algorithm can select an intervention that is expected to reduce the non-distinctiveness of the agent's path, even if the intervention does not necessarily reduce wcd.

2.3 Active Observers

There is a growing body of work on observers who use agency in the domain to help reveal the subject agent's goal more quickly. Kabanza *et al.* [2010] and Bisson *et al.* [2011] explore 'provoking actions' to force opponents to reveal their intentions through their reactions. These works depend on direct cause-effect relationships between provoking actions and opponent reactions. By contrast, our work allows observers to plan longer-term sequences of interventions without immediate effect. Mirsky *et al.* [2018] model plans as context-free grammars with nonterminals that can be refined via productions into primitive actions. They query the subject online about whether a coarse plan can be refined to the correct hypothesis and can thus prune significant portions of the observer's hypothesis set. However, their observer does not have agency to act in the world.

Shvo and McIlraith [2020] introduce Active Goal Recognition, where the observer is granted agency to act within the domain. The work focuses on partially observable domains, so they limit the observer's plans to facilitating observations that do not alter any of the subject's possible plans. They choose interventions to maximally reduce the size of their goal hypothesis prior to the next observation of the subject. Our work proposes a more complete formulation where the observer is allowed to invalidate possible subject plans and execution of observer and subject actions are interleaved.

Goal Elicitation Planning [Amos-Binks and Cardona-Rivera, 2020] defines a setting where the observer's interventions are interleaved with the subject's actions as the observer seeks to minimize wcd online. From the start configuration, they generate a set of possible plans for the observer, then select from this set as the subject begins executing its own plan. However, once committed to a plan, their formulation does not permit an observer to react to actions the subject takes, even if the subject transitions to a part of the state space unaffected by the observer's plan. Moreover, once an observer plan finishes executing, it has no means of generating a new plan to further reduce wcd. By contrast, our formulation allows the observer to react to subject actions and adjust its plan accordingly.

CHAPTER 3
GRD in Online Settings

Existing work on GRD optimizes distinctiveness in the form of wcd or ecd. In this chapter, we analyze these metrics and identify problematic cases. We then discuss an alternative that has advantages in online settings. We also emphasize the importance of modeling the passage of time.

3.1 Objective Functions

Intuitively, a goal recognition system should report the correct goal of a subject quickly enough that the observer has time to react to this information. Distinctiveness, the underlying metric of wcd and ecd, focuses solely on how quickly goals can be identified. We will contrast this with ψ, an objective that incorporates the reaction time an observer will have to use the information about the subject's goal.

3.1.1 Distinctiveness

One underlying problem with wcd as an objective function is that its focus on the worst case can cause useful interventions to be overlooked. Consider Figure 3.1, which represents a deterministic domain where the wcd equals the optimal cost to goal A. There are two plan prefixes whose length equals wcd: one that leads to both B and C and becomes distinct at time step $wcd + 1$, and one that leads to goal B and passes through goal A at time step wcd, also becoming distinct at $wcd + 1$. The red circle indicates an intervention that eliminates the plan to B that passes through A. However, the standard GRD objective would ignore

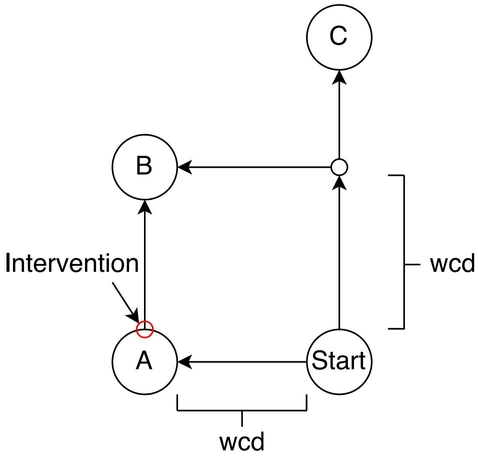

Figure 3.1: wcd misses the intervention

this intervention since it does not affect wcd. Clearly if the agent's goal is A, it is preferable to distinguish it even if the distinctiveness of B and C cannot be reduced.

To address this issue, we look to expected case distinctiveness (ecd) [Wayllace *et al.*, 2017]. ecd is a function which takes a state s in a Stochastic Shortest Path Markov Decision Process (SSP-MDP) and returns the expected length of the subject's non-distinct path prefix through the MDP. To find $ecd(s)$, they recursively find ecd of all the successors of s and weight each of them by the transition probabilities of actions leading to the successor and the actions' likelihood w.r.t. a set of given prior probabilities over possible goals. The base case of ecd is a state for which the goal is uniquely identifiable. Though ecd was designed for stochastic domains, determinism is merely a special case. Let us weight each goal in Figure 3.1 equally so that we can compute ecd for the start state. Each successor of the start state leads to a branch that becomes distinct at $wcd + 1$, and each goal has equal probability, so before executing the intervention, $ecd(Start)$ can be computed by taking the weighted average of the distinctiveness over all goals: $ecd(Start) = \frac{wcd}{3} + \frac{wcd}{3} + \frac{wcd}{3} = wcd$. It is clear that by executing the intervention, $ecd(Start) = 0 + \frac{wcd}{3} + \frac{wcd}{3} = \frac{2}{3}wcd$ since goal A becomes distinct after taking a single step, thus identifying that the intervention is worth executing.

However, ecd still has a key flaw: as a measure based on distinctiveness, it is insensitive

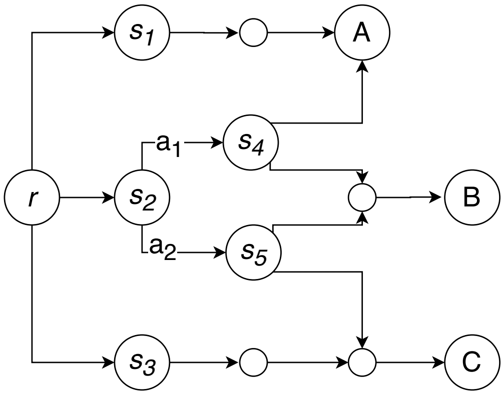

Figure 3.2: ecd provides no guidance

to the time remaining after the goal is identified, the reaction time. Consider Figure 3.2, in which there are three possible goals with a priori equal probability: A, B, and C. All arrows describe deterministic actions of cost 1. There are two possible interventions: removing a_1 or a_2. It is possible that an agent pursuing goal A will not reveal their goal until it is achieved by taking the path $\langle r, s_2, s_4, A\rangle$. The only way to guarantee that all goals will be identified before they are achieved is to remove action a_1. However, removing either a_1 or a_2 will have identical effects on ecd since their removal is symmetrical. A plan to goal A via s_4 is distinct after 3 steps, which is the same as a plan to C via s_5, despite the fact that A is achieved in 3 steps where C is not. Therefore, a_1 will not be identified as a more urgent intervention than a_2 despite our intuition that a_1 is better. In offline GRD with budget constraints, a more sophisticated prioritization is important for executing the most impactful interventions with limited resources. When intervening online, prioritization is even more critical since we may not have time to execute all impactful interventions prior to the subject transitioning beyond them.

3.1.2 ψ: Fraction of the subject's plan

In their work on Active Goal Recognition, Shvo and McIlraith [2020] compute a different objective: the fraction of the subject's plan that is completed prior to the goal being recognized. Their approach does not directly minimize this metric, but its use is indicative of the

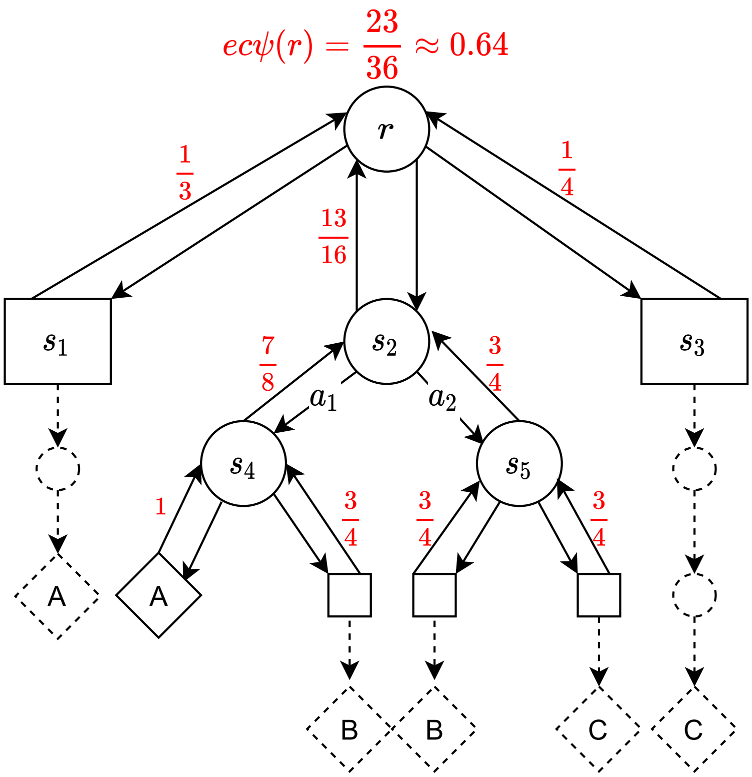

Figure 3.3: Tree for calculating $ec\psi$ on root node.

properties of a useful online detection system, e.g. one that considers the reaction time the observer will have to utilize the information once the goal is identified. Pursuing this idea, let us formally define this metric.

Definition 1 $\psi : S \times \mathcal{G} \rightarrow [0, 1]$ *is a function that takes a state and a goal and returns the fraction of the plan that has been executed to achieve that goal relative to a start state s_{root}:*

$$\psi(s, g) = \frac{|\textsc{OptimalPlanToState}(s_{root}, s)|}{|\textsc{OptimalPlanToGoal}(s_{root}, g)|} \ . \tag{3.1}$$

We will clarify all of our assumptions in Section 4.1. For the purpose of this argument, we use plan length as a measure of time and so use the fraction of the executed plan length over the total plan length. Let $ec\psi$ be a function that behaves exactly as ecd except, instead of returning the expected distinctiveness of a state, it returns the expectation of ψ for the first state where we can identify the subject's true goal. As with ecd, lower numbers are better.

Returning to the example from Figure 3.2, if we replace ecd with $ec\psi$, it becomes clear that removing a_1 is the best intervention. Figure 3.3 shows the tree induced by finding $ec\psi$ of the root state r. Goals are represented as diamonds. Leaf nodes are those for which the subject could only be pursuing one goal, and are represented as rectangles, except where a leaf node is a goal. For clarity, we represent transitions the subject would execute after their goal is distinct with dotted lines, however these nodes are not traversed in the tree to calculate $ec\psi$. The value of a leaf node representing state s is $\psi(s)$, and the values of non-leaf nodes are the average values of its children weighted by the probability of the action that leads to that each child, which depends on the goal priors for the goals that child could lead to. For example, children of the root node r have values $\frac{1}{3}$, $\frac{13}{16}$, and $\frac{1}{4}$ for successors s_1, s_2, and s_3 respectively. Again, we assume all goals are equally likely, so the total probability of the subject transitioning to s_1 is $\frac{1}{6}$ because there is a $\frac{1}{3}$ probability the subject is pursuing goal A, and if the subject is pursuing goal A, there is a $\frac{1}{2}$ probability they will transition to s_1, so $\frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$. Accordingly, the respective action probabilities for s_2 and s_3 are $\frac{2}{3}$ and $\frac{1}{6}$. Therefore, the value of $ec\psi(r) = \frac{1}{3} \times \frac{1}{6} + \frac{13}{16} \times \frac{2}{3} + \frac{1}{4} \times \frac{1}{6} = \frac{23}{36} \approx 0.64$.

Observe that for a subject pursuing B, the effect of removing a_1 or a_2 is identical, so let us isolate the effect of these interventions for cases where the subject is pursuing either goal A or C. We can decompose $ec\psi = p_A \times ec\psi_A + p_B \times ec\psi_B + p_C \times ec\psi_C$ where p_X is the probability of Goal X, and $ec\psi_X$ is the expected ψ for instances where the subject's goal is actually revealed to be Goal X. Since all goals in our example are equally likely, we must simply prove that removing a_1 has larger effect on $ec\psi_A$ than removing a_2 has on $ec\psi_C$.

Figure 3.4 shows the reduced tree corresponding to $ec\psi_A$. The branch of the tree where the subject transitions toward Goal B is faded since a subject whose ground truth goal is A would not pursue that branch, but we leave it in the figure so that it is clear when the subject's goal actually becomes distinct. A subject pursuing A could transition to s_1 in which case their plan is distinct after 1 step, so $ec\psi_A(s_1) = \frac{1}{3}$. If action a_1 is still available, the subject could also transition to s_2 in which case the only remaining plan to A is not

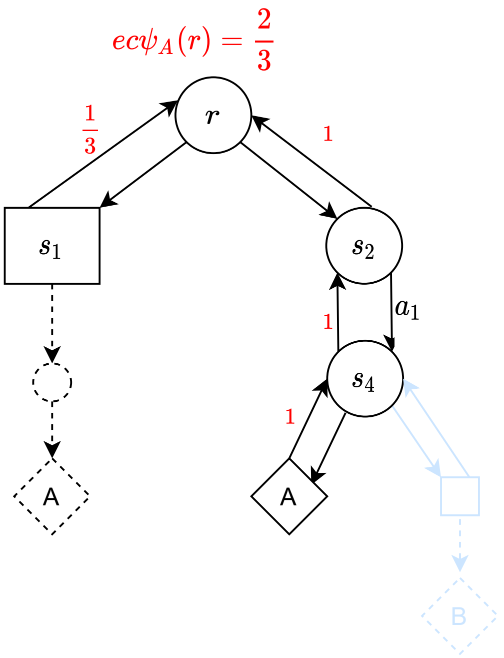

Figure 3.4: Tree isolating instances where Goal A is revealed to be the actual subject goal

distinct from a plan to B until A is achieved, i.e. $ec\psi_A(s_2) = 1$. Assuming that each action that leads to the subject's goal is equally likely, $ec\psi_A(r) = \frac{1}{2} \times \frac{1}{3} + \frac{1}{2} \times 1 = \frac{2}{3}$ If we remove a_1, then the plan that is distinct after one step is the only one remaining, so $ec\psi_A(r) = \frac{1}{3}$, a reduction of $\frac{1}{3}$. Repeating this for a subject pursuing C, prior to any interventions we have $ec\psi_C(r) = \frac{1}{2} \times \frac{1}{4} + \frac{1}{2} \times \frac{3}{4} = \frac{1}{2}$. After removing a_2, $ec\psi_C(r) = \frac{1}{4}$, a reduction of $\frac{1}{4}$. Since the reduction from removing a_1 ($\frac{1}{3}$) is greater than the reduction from removing a_2 ($\frac{1}{4}$), minimizing $ec\psi$ would prioritize a_1 for removal.

$ec\psi$ effectively normalizes both distinctiveness and reaction time across all goals so that the urgency of distinguishing a goal is inversely proportional to the length of the optimal plan to achieve it. In other words, close goals are prioritized over distant goals. The normalizing effect of ψ provides guidance when interventions have effects of similar magnitude. More weight is given to those that affect close goals because a reduction in time to identify the goal accounts for a larger fraction of the plan to a close goal than a distant one. This guidance is important in online settings with time pressure where we may not have time to execute all meaningful interventions, and so must prioritize them.

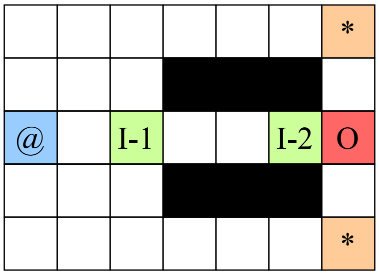

Figure 3.5: Temporal reasoning in AGRD.

3.1.3 Choosing Objectives

Ultimately, the choice of objective is domain-specific. Consider a wildlife photographer who wishes to photograph a rare animal and knows that they can be found at one of a few streams at certain times of day. They set up mics and low-fidelity cameras in the environment that allow for basic tracking, but to get the Pulitzer Prize shot, they need the stream. Perhaps the photographer knows that if the animal senses them, it will run off to drink at another stream, so they need as much time as possible to set up at the correct stream so that they can be well hidden by the time the animal arrives. One can imagine various interventions to encourage the animal to choose a direction, for instance by setting up a barrier at some common trail to force the animal to take a different, more distinct path to one of the streams. Distinguishing close goals may not be a priority in this case because, if there is not enough time to set up, then the photographer will not get the shot anyway. In this case, maximizing the reaction time is paramount, no matter how close or far the distinguished goal is. In Chapter 4, we will define the objective function of our problem setting generically to accommodate varying requirements appropriate to different domains.

3.2 Temporal Reasoning

Our consideration of reaction time in the formulation of ψ illustrates a gap in the conventional Active Goal Recognition setting: a formal treatment of the passage of time. If we wish to minimize a metric based on ψ, we need to be able to execute interventions that are still relevant by the time they are carried out. For example, consider Figure 3.5. In this simple Grid-World with 4-way movement, the subject agent (blue) has one of two goals (orange). The observer agent (red) may block any green cell it is adjacent to. The observer follows the same movement rules as the subject, and they can occupy the same cell. We assume the subject follows optimal plans and the observer must not change the optimal cost to any goal. By any of the discussed objectives, it is clear that blocking *I-1* would force the subject to reveal its goal more quickly than *I-2*, since the subject must turn left or right after one step. However, under a realistic online model where the subject's and observer's actions are interleaved, the observer agent does not have enough time to reach and block *I-1* before the subject enters that cell. An observer reasoning with temporal awareness will identify that *I-2* is actually the optimal intervention since the subject would then turn away from the corridor at the latest after they reach cell *I-1*.

CHAPTER 4

Active Goal Recognition Design

We now propose a new problem setting that combines elements of prior models and applies them to the online setting to explicitly model the passage of time and enable the consideration of sequences of interventions.

4.1 Preliminaries

We assume action outcomes are deterministic and that the subject and observer take turns executing actions. Both agents immediately observe the effects of the other's action prior to selecting their next action. All actions take fixed time to execute, though costs can vary. We assume the subject agent acts optimally with respect to the domain costs C^* (defined formally in Section 4.2) and that it is unaware of or agnostic to the observer. This means the subject is not antagonistic to the observer by attempting to disrupt or hinder the goal recognition process through purposeful choice of maximally non-distinct plan prefixes. It also means that the subject's planning is not strategic, i.e. it is not affected by interventions the observer has not yet taken. Because we assume the subject is non-strategic, we will also assume that, given a goal g, all optimal plans the subject could execute to achieve g are equally likely.

We assume that the observer is neither assisting nor impeding the subject, and thus it must not change the optimal cost to achieve any goal. The observer's interventions can, however, remove optimal plans available to the subject as long as the new system state includes at least one plan for each goal the subject may be pursuing whose cost equals the remaining cost-to-go of the optimal plan(s) the observer invalidated. Accordingly, we assume that if the plan the subject was executing is invalidated by an intervention, it is immediately able to select a new optimal plan to its goal.

4.2 Problem Formulation

Definition 2 *An AGRD domain is a tuple $D = \langle S, A_{subj}, A_{obs} \rangle$ where S is a set of system states representing the subject, observer, and their environment and A_{subj} and A_{obs} are functions that, given state s, return the actions available in s to the subject and observer, respectively. An action $a \in A(s)$ is a function that maps a state to its successor and the positive non-zero cost $c(a(s))$ of that transition.*

An action that maps a state to itself is called an identity action and can be used by the observer when it does not have a more useful action to execute. We refer to actions returned by A_{obs}, i.e. observer actions, as interventions.

Definition 3 *An AGRD Problem Instance is a tuple $I = \langle D, s_{root}, \mathcal{G}, P_{\mathcal{G}}, \rho \rangle$ where D is an AGRD domain, s_{root} is the start state, $\mathcal{G} = \{g_1, ..., g_n\}$ is the set of possible goals the subject could be pursuing, where a goal $g \in \mathcal{G}$ is a set of states, $P_{\mathcal{G}} = \{p_1, ..., p_n\}$ where p_i is the prior probability for $g_i \in \mathcal{G}$, and ρ is an objective function mapping a state to a real number.*

Definition 4 *An action sequence $\alpha_{s_0} \in \mathbb{A}_{s_0} = \langle a^1_{obs}, a^1_{subj}, ..., a^n_{obs}, a^n_{subj} \rangle$ is a sequence of alternating interventions and actions starting from state s_0 such that $s'_i = a^i_{obj}(s_{i-1})$, $s_i = a^i_{subj}(s'_i)$, $a^i_{obj} \in A_{obj}(s_{i-1})$, and $a^i_{subj} \in A_{subj}(s'_{i-1})$. $\mathbb{A}_{s_0}$ refers to the space of all possible action sequences rooted at s_0, which represents all possible evolutions of the system.*

α describes the interleaved action execution of both the observer and subject as the system evolves. We define the space of all possible action sequences as $\mathbb{A}$. Action sequences are different from the plans entertained by the subject, which do not include interventions. In a slight abuse of notation, we will refer to the accumulated cost incurred by subject actions along an action sequence from s_{root} to s as $c_{subj}(s)$ and the number of subject actions in the sequence as $length_{subj}(s)$.

Definition 5 *A subject plan $\phi_{s_0,g} = \langle a^1, ..., a^n \rangle$ is a sequence of actions starting from state s_0 such that $s_i = a^i(s_{i-1})$, $a^i \in A_{subj}(s_{i-1})$, and $s_n \in g$.*

We will use this concept to define the behavior of an optimal subject which, according to our assumptions, is ignorant of the observer.

Definition 6 *For a problem instance I with start state s_{root}, $C^*(g)$ is the minimum cost among all subject plans $\phi_{s_{root},g}$ and we notate a plan of this cost as $\phi^*_{s_{root},g}$.*

Because we assume the observer cannot change the optimal cost to any goal, $C^*(g)$ always refers to the original optimal cost of g. The optimal cost to g from an intermediate state s along an optimal path to g is $C^*(g) - c_{subj}(s)$, which we will denote as $C^*(s, g)$

Definition 7 *H_s is the set of goals g that the subject could be pursuing.*

We refer to sets H as goal hypotheses. They represent the remaining goals an optimal subject could be pursuing. We describe how goals are pruned from H in Section 4.3.1.

Definition 8 *A solution to an AGRD instance I is a policy $\pi : S \rightarrow A_{obs}$ that maps all states s reachable via an action sequence $\alpha_{s_{root}}$ to an intervention a_{obs}.*

For any intervention $a_{obs}(s) = s'$, any subject plans from s may be invalidated as long as $\forall g \in H_s, C^*(s, g) = C^*(s', g)$. In other words, an intervention cannot change the optimal cost to achieve any goal that is in the goal hypothesis of the original state.

4.3 Defining Optimal Solutions

We define optimal solutions in terms of a generic objective function $\rho : S \rightarrow \mathbb{R}$ that maps a state to a real number. We maximize the expected value of ρ over a state space tree formed

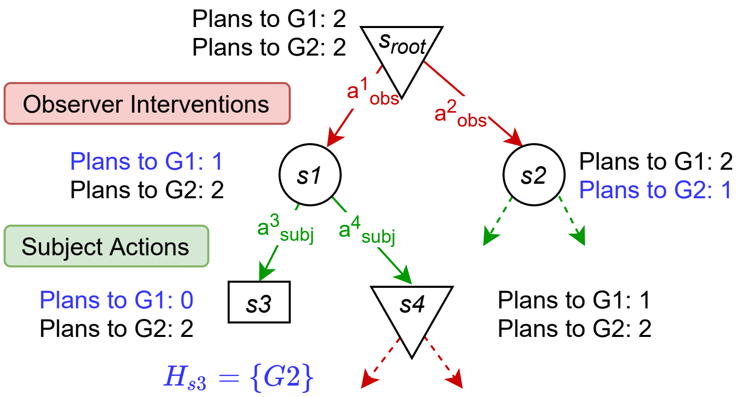

Figure 4.1: Example state space tree induced by $e_\rho(s_{root})$.

by simulating interventions and subject actions; see Figure 4.1 for an example. At any state, there exists some number of optimal subject plans to each possible goal. Each intervention may change the available optimal plans by invalidating or enabling subject plans. In the figure, the intervention a^1_{obs} invalidates one of the two plans to goal *G1* so that there is only one plan left to *G1* in the state s_1. Each subject action may also reduce the number of optimal plans if the subject transitions beyond non-distinct plan prefixes. In the figure, a subject that takes a^3_{subj} transitions beyond the final non-distinct prefix that could still lead to *G1*. In other words, the number of subject plans available to goal $G1$ in s_3 is 0, meaning $G1$ can be pruned from the goal hypothesis for s_3. Since $|H_{s_3}| = 1$, s_3 is a leaf node of the tree whose value will be defined as $\rho(s_3)$. The value of non-leaf nodes is computed by backing up the value of child nodes to their parents. We take the minimum over possible interventions since we are looking for interventions that minimize our objective, but we must take the expectation over possible subject actions (weighted, as we explain below, by conditional goal probability) because we assume a non-strategic subject. This is an expectimin tree (expectation nodes depicted as circles, min as triangles) in contrast to the more common two-player zero-sum game minimax tree.

Definition 9 *Let P_s be a probability distribution over all actions the subject could take in state s, representing the probability the subject will select the corresponding action.*

We will formally derive P_s and our method for pruning goals from H in Section 4.3.1.

Definition 10 *$e_\rho : S \to \mathbb{R}$ is a function that computes the expectation of the best achievable ρ of a given state for a given scoring function ρ:*

$$e_\rho(s) = \begin{cases} \rho(s) & \textit{if } |H_s| \leq 1 \\ \underset{a_{obs} \in A_{obs}(s)}{min} \; score\big(a_{obs}(s)\big) & \textit{otherwise} \end{cases} \tag{4.1}$$

$$score(s) = \underset{a_{subj} \sim P_s}{\mathbb{E}} \Big[e_\rho\big(a_{subj}(s)\big) \Big] \; . \tag{4.2}$$

The mutual recursion among equations 4.1 and 4.2 yields the alternating layers of the state space tree as in Figure 4.1. Each leaf of this tree corresponds to a unique action sequence $\alpha \in \mathbb{A}$, or in other words, one unique evolution of the system.

We now define an optimal intervention in terms of e_ρ and *score*. Given a problem instance I, an optimal policy π^* returns the intervention a^*_{obs} that is expected to lead to the lowest score:

$$a^*_{obs} = \underset{a \in A_{obs}(s)}{\operatorname{argmin}} \; score(a(s)) \; . \tag{4.3}$$

A perfectly rational observer in some state will execute the intervention that is expected to minimize the penalty value ρ when the goal is revealed. According to our assumptions, the subject does not act strategically or in response to the observer's objective. Therefore, the expectation is distributed by our prior belief as action probabilities P_s, which are computed using Bayesian updates as the system evolves as we discuss next. e_ρ and *score* encode these assumptions allowing us to produce the optimal intervention.

4.3.1 Goal Probabilities

The probability distribution P_s over subject actions in a state s depends on the probability distribution over the goals it might be pursuing from that state. At s_{root}, these are $P_{\mathcal{G}}$. As the subject acts to achieve its goal, some goals may become inconsistent with the subject's executed plan, requiring us to compute a goal posterior describing the updated probabilities over remaining goals given our observations. We denote these posteriors as P_{H_s}, referring to the probability of each goal in the goal hypothesis H_s for the state s. We compute P_{H_s} via a sequence of Bayesian updates where the goal posterior of each update is used as the prior for the next.

Let $P_{H_s}(g_i)$ be the prior probability of g_i before some action a is executed in state s, and let $Plans_i(s)$ return all optimal subject plans ϕ^*_{s,g_i}. From our assumption that all optimal plans to a given goal are equally likely, the likelihood $P_s(a|g_i)$, the probability of subject action a given g_i, is the fraction of optimal plans to g_i for which a is the next action:

$$P_s(a|g_i) = \frac{|Plans_i(a(s))|}{|Plans_i(s)|} \ . \tag{4.4}$$

Then the marginal likelihood $P_s(a)$, the probability of a, is

$$P_s(a) = \sum_{g_i \in H_s} P_{H_s}(g_i) P_s(a|g_i) \tag{4.5}$$

and Bayes' rule tells us the posterior $P_{H_s}(g_i|a)$, the probability of the goal g_i given an action a such that $a(s) = s'$, is

$$P_{H_{s'}}(g_i) = P_{H_s}(g_i|a) = \frac{P_{H_s}(g_i) P_s(a|g_i)}{P_s(a)} \ . \tag{4.6}$$

In other words, $P_{H_{s'}}(g_i) \propto P_{H_s}(g_i) P_s(a|g_i)$. The normalization factor is the sum of the action likelihood times goal prior over all possible goals (Eq. 4.5).

Eq. 4.6 is computed over all goals to get the full posterior $P_{H_{s'}}$. When we receive our next observation, $P_{H_{s'}}$ will be used as the prior in the update for that observation. If the subject transitions to a successor that is not on any optimal plan to some goal g_i, $P_s(a|g_i)$, and thus the posterior $P_{H_{s'}}(g_i)$, becomes 0, and g_i is pruned from the hypothesis $H_{s'}$.

4.3.2 Objectives ρ

The metrics discussed in Chapter 3 can be defined as alternative functions ρ.

Definition 11 *Function $\rho^{\psi} : S \rightarrow [0, 1]$ returns the fraction of the subject's plan that has been executed:*

$$\rho^{\psi}(s) = \begin{cases} \frac{length_{subj}(s)}{length_{subj}(s)+|\phi^{*}_{s,g}|} & \textit{if } |H_s| = |\{g\}| = 1 \\ 0 & \textit{otherwise .} \end{cases} \tag{4.7}$$

Note that in Definition 3.1, ψ is initially defined assuming a stable domain that does not alter with the progression of the system. Equation 4.7 is defined in terms that are resilient to changes in the domain that may, for instance, invalidate the original subject plan $\phi^{*}_{s_{root},g}$. This redefinition of ψ ultimately maintains the same semantics: the fraction of the subject's plan that has been executed to that state. Minimizing $e_{\rho^{\psi}}$ within our optimal solution prioritizes distinguishing goals that will be achieved more quickly, thus preventing goals from being achieved before they are distinguished.

To see how wcd and ecd relate to our approach, we can re-frame distinctiveness:

Definition 12 *Function $\rho^{d} : S \rightarrow \mathbb{R}^{\geq 0}$ returns the length of the action sequence the subject has executed:*

$$\rho^{d}(s) = \begin{cases} length_{subj}(s) & \textit{if } |H_s| = 1 \\ 0 & \textit{otherwise .} \end{cases} \tag{4.8}$$

Here again, Equation 4.8 defines distinctiveness not in terms of a subject plan $\phi^{*}_{s_{root},s}$ that may no longer exist, but using the notation introduced to describe action sequences α. Using $e_{\rho^{d}}$ approximately minimizes ecd, not wcd, since we are using an expectation instead of a global value. $e_{\rho^{d}}$ and ecd are not exactly equivalent because of differences in how the action probability is computed, namely that $e_{\rho^{d}}$ considers the number of possible plans the subject could be pursuing whereas ecd considers only the relative size of goal hypotheses between actions. Nevertheless, using ρ^{d}, our formulation can optimize distinctiveness if an application requires it.

4.4 Analysis

We establish properties of an optimal solution by analyzing the tree induced by e_ρ. Let T_{e_ρ} refer to this tree.

Lemma 1 *Given any state s and an optimal subject action $a_{subj} \in A_{subj}(s)$ with $a_{subj}(s) = s'$, $\forall g \in H_{s'}$, $C^*(s,g) > C^*(s',g)$.*

Proof: According to Definition 2, action costs are > 0, therefore $c_{subj}(s) < c_{subj}(s')$. An optimal subject must always choose an action that reduces the remaining cost to achieve their goal by exactly the cost of the action, or they are not acting optimally. The goal hypothesis is the set of goals that an optimal subject could be pursuing at some state, meaning every action taken has reduced the optimal cost to achieve every goal in the remaining goal hypothesis. Therefore, the remaining cost to achieve any goal in $H_{s'}$ is monotonically decreasing and not equal to its predecessor. □

Lemma 2 *Given any two states s and s_a such that the node in T_{e_ρ} representing s_a is an ancestor of the node representing s, $\forall g \in H_s$, $C^*(s_a, g) \geq C^*(s,g)$.*

Proof: First note that for any successor $s' = a_{obs}(s)$, our neutral observer assumption requires that the remaining costs to achieve goals in the goal hypothesis remain the same. i.e. $\forall g \in H_s$, $C^*(s,g) = C^*(s',g)$.

Lemma 1 tells us that for any successor of a subject action $s' = a_{subj}(s)$, $\forall g \in H_{s'}$, $C^*(s,g) > C^*(s',g)$. We can therefore say more generally that for any successor s', $\forall g \in H_{s'}$, $C^*(s,g) \geq C^*(s',g)$. This statement describes a relationship between parent and direct successor, however we can trivially generalize it to any ancestor by the transitive property of inequalities. □

Theorem 1 *T_{e_ρ} is a Directed Acyclic Graph (DAG).*

Proof: We must prove that no node of T_{e_ρ} has an edge to an ancestor node or itself.

We begin by proving that edges defined by subject action transitions exhibit this property. Per Lemma 1, for any successor s' of a subject action, $C^*(s', g)$ is monotonically decreasing for all goals in the hypothesis $H_{s'}$. If the successor s' were also an ancestor of the state s, or if $s = s'$, Lemma 2 would imply $\forall g \in H_{s'}$, $C^*(s, g) \leq C^*(s', g)$, which contradicts Lemma 1.

We now prove that edges defined by observer interventions also exhibit this property. Let s now refer to the parent state of a transition defined by a_{obs}. First note that due to the interleaving of observer and subject actions, observer interventions are executed on either the root of the tree or successors of subject actions. Applying Lemmas 1 and 2, we can say that for all ancestors s_a of s, $\forall g \in H_s$, $C^*(s_a, g) > C^*(s, g)$. According to our neutral observer assumption, the observer cannot execute an intervention that changes the optimal cost to any goal. Therefore, we cannot transition to any ancestor of the current state. The observer can, however, execute an identity intervention that does not change the system state, i.e. $s' = s$. However, the successor node of an identity intervention is not the same node of T_{e_ρ} as its predecessor, despite representing the same underlying state, because the successors of s' are defined by subject actions $A_{subj}(s')$, not observer interventions. As noted above, the subject will act to reduce the remaining cost to its goal, therefore preventing a return to any ancestor. □

While our formulation does not require that subject actions have a fixed minimum cost, for the remainder of our analysis, let us make the reasonable assumption that one exists:

$$\lfloor c \rfloor = \min_{a \in A_{subj}, s \in S} c(a(s)) \ . \tag{4.9}$$

Let us also assume that A_{subj} and A_{obs} return finite sets of actions for any one state. We define our maximum branching factor accordingly:

$$b = \max_{A \in \{A_{subj}, A_{obj}\}, s \in S} |A(s)| \ . \tag{4.10}$$

Theorem 2 *The size of T_{e_ρ} is finite.*

Proof: Lemma 1 establishes that every action the subject takes monotonically reduces the remaining cost to achieve all goals in the successor's goal hypothesis. Since the minimal cost of any action is $\lfloor c \rfloor$, we can trivially upper-bound the depth of T_{e_ρ}. To avoid confusion with notation of distinctiveness using d, we refer to the depth bound of T_{e_ρ} as κ:

$$\kappa \leq 2 * \max_{g \in \mathcal{G}} \frac{C^*(g)}{\lfloor c \rfloor} \ . \tag{4.11}$$

If every action the subject executes has cost $\lfloor c \rfloor$, then the subject can take at most a number of actions equal to the cost of the most expensive goal divided by $\lfloor c \rfloor$. Since the subject and observer interleave actions, the tree depth can be bounded by twice the maximum number of actions a subject can take. We have established that the depth κ and the branching factor b are both finite. Therefore, T_{e_ρ} must be finite. □

Theorem 2 establishes that κ is bounded by the cost of the most expensive goal. However, leaf nodes of T_{e_ρ} are defined as those for which $|H_s| \leq 1$ (Eq. 4.1). The tree does not descend down to states where the subject has achieved their goal if the goal becomes distinct prior to achievement.

Theorem 3 *Let*

$$\dot{g} = \operatorname*{argmax}_{g \in \mathcal{G}} C^*(g) \qquad \textit{Most expensive goal}$$

$$\ddot{g} = \operatorname*{argmax}_{g \in \mathcal{G}/\{\dot{g}\}} C^*(g) \qquad \textit{Second most expensive goal}$$

Then the depth of T_{e_ρ} is bounded by the second most expensive goal $\ddot{g} \in \mathcal{G}$:

$$\kappa \leq 2 * \frac{C^*(\ddot{g})}{\lfloor c \rfloor} \ . \tag{4.12}$$

Proof: Let κ^1 and κ^2 refer to the maximum number of actions that can be taken prior to a subject achieving the most and second most expensive goals, respectively:

$$\kappa^1 = 2 * \frac{C^*(\dot{g})}{\lfloor c \rfloor}$$
$$\kappa^2 = 2 * \frac{C^*(\ddot{g})}{\lfloor c \rfloor}$$
$$\kappa^1 \geq \kappa^2 \ . \tag{4.13}$$

After κ^2 actions have been taken, the subject is guaranteed to have achieved $\ddot{g}$ if that was their goal. If $\ddot{g}$ was not their goal, after κ^2 actions there can be at most one goal that has not been achieved: $\dot{g}$. Thus, the size of the hypothesis will be 1, it is a leaf node, and the tree does not expand beyond it making the depth bound $\kappa = \kappa^2$. □

Corollary 1 *The size complexity of T_{e_ρ} is bounded by $O(b^\kappa)$.*

Proof: This follows from the definition of b and κ as applied to a tree graph structure. □

We conclude our analysis by considering the optimal substructure of T_{e_ρ}. Problems that exhibit optimal substructure are those where the optimal solution contains optimal solutions to its subproblems [Cormen *et al.*, 2009]. This property can be leveraged to speed up computation with dynamic programming solutions. T_{e_ρ} has this property due to the recursive definition of e_ρ, but let us formalize this.

Theorem 4 *T_{e_ρ} exhibits optimal substructure.*

Proof: We assert both that branches defined by the expectation over subject actions and those defined by the minimum over observer interventions have optimal substructure. We will prove both by contradiction. Let us assume that the optimal solution at a branch rooted in some node n includes at least one suboptimal value from among its children.

Subject Action Nodes: Let n be a node of T_{e_ρ} whose value is the expectation over children defined by subject actions. Subtrees induced by each subject action are considered with weight according to the action's marginal likelihood, and we need only consider subtrees where the likelihood is greater than 0. A suboptimal value for a child representing some state s of a subject action node is one that is greater than $e_\rho(s)$ (Eq. 4.1). If any value used in the expectation over subtrees of n are sub-optimal, then the value of n could be improved (i.e. reduced) by instead using the optimal value, computed with e_ρ, of the child node. This contradicts our assumption, so subject action nodes must exhibit optimal substructure.

Observer Intervention Nodes: Let n be a node of T_{e_ρ} whose value is the minimum of its children over all possible interventions. A suboptimal value for a child representing state s is one that is greater than $score(s)$ (Eq. 4.2). Our assumption requires that the value of n be derived from a suboptimal value of one of its children. However, we could further reduce this value by instead using $score$. This contradicts our assumption, so observer intervention nodes must exhibit optimal substructure. □

At first glance, this appears to be helpful in exploring T_{e_ρ}, since dynamic programming algorithms can take advantage of optimal substructure to save computation by preventing the exploration of duplicate sub-trees. However, recall that T_{e_ρ} implicitly captures the history of the subject's and observer's actions through the incremental computation of P_s via Bayesian updates. As a consequence, nodes of T_{e_ρ} that share the same system state s may have different posteriors $P^i_{H_s}$ where $P^i_{H_s}$ is one of many possible goal posteriors for state s whose values depend on the specific action sequence α corresponding to the branch of T_{e_ρ} the subject and observer have traversed.

Consider Figure 4.2, which represents a progression of states through a Grid-World domain with identical rules as Figure 3.5. Here g_1 and g_2 refer to the subject's possible goals. At the start state s_{root}, the observer can block cells *I-1* and *I-2* in either order before the subject reaches cell *C-1*. s_2 refers to the system state where the subject is located in *C-1* and both *I-1* and *I-2* are blocked. A subject in s_2 must take an action that distinguishes their goal, but the order in which *I-1* and *I-2* are blocked is not inherently described by the state; s_2 is identical in either case. However, the order the interventions are executed changes the resulting goal posteriors of s_2. Assume both goals are equally likely. In the figure, the

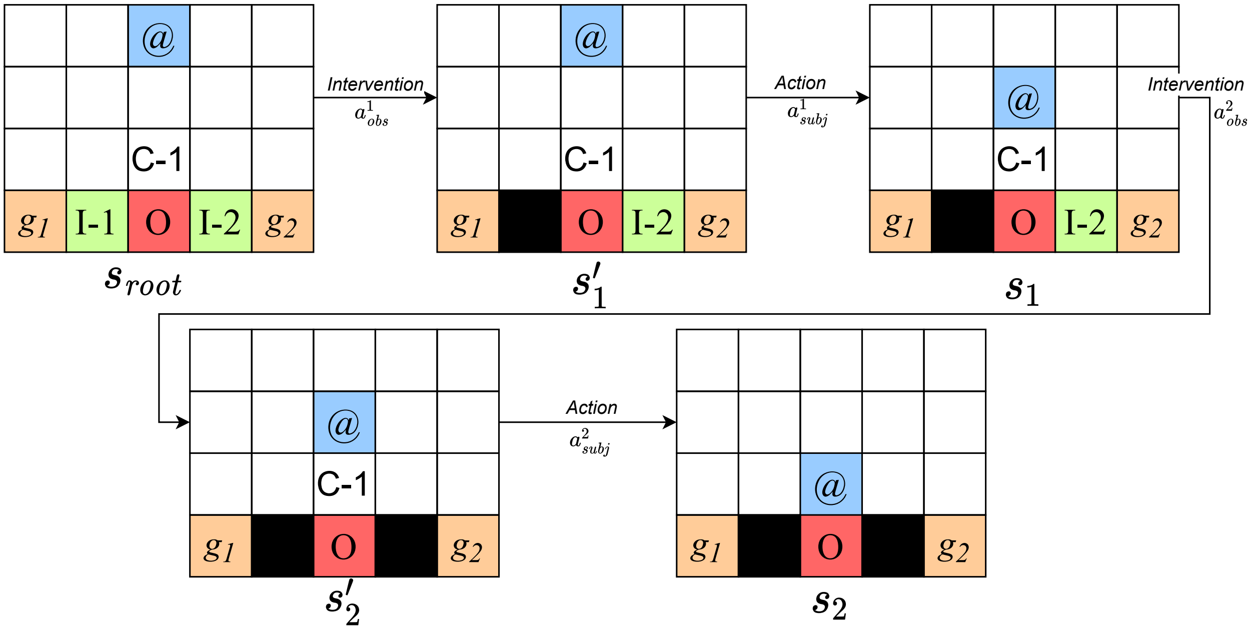

Figure 4.2: The goal posteriors would be calculated differently with a different ordering of interventions

observer blocks *I-1* first (state s'_1). From the subject's location in s'_1, there remain 6 plans available to achieve g_1, and 10 plans for g_2. If the subject moves toward *C-1* via action a^1_{subj}, only 3 plans to g_1 and 6 plans to g_2 would remain. The goal posteriors for the observation of a^1_{subj} would be computed as:

$$\text{Action Likelihoods} \qquad P_{s'_1}(a^1_{subj}|g_1) = \frac{3}{6} = \frac{1}{2}$$

$$P_{s'_1}(a^1_{subj}|g_2) = \frac{6}{10} = \frac{3}{5}$$

$$\text{Marginal Likelihood} \qquad P_{s'_1}(a^1_{subj}) = \frac{1}{2}\left(\frac{1}{2} + \frac{3}{5}\right) = \frac{11}{20}$$

$$\text{Bayesian updates} \qquad p^1_1 = \frac{\frac{1}{2} \times \frac{1}{2}}{\frac{11}{20}} = \frac{5}{11} \tag{4.14}$$

$$p^1_2 = \frac{\frac{1}{2} \times \frac{3}{5}}{\frac{11}{20}} = \frac{6}{11} \ . \tag{4.15}$$

Now when the observer blocks *I-2* (state s'_2), the symmetry of the domain is restored and only 3 subject plans remain to either goal. If the subject moves into *C-1* (state s_2), there will be only 1 plan for each goal (going left for g_1 or right for g_2), and the resulting posteriors will be unchanged in this transition since the likelihood of that action given either goal is the same:

$$\begin{align}
\text{Action Likelihoods} \qquad & P_{s_2'}(a^2_{subj}|g_1) = \frac{1}{3} \nonumber \\
& P_{s_2'}(a^2_{subj}|g_2) = \frac{1}{3} \nonumber \\
\text{Marginal Likelihood} \qquad & P_{s_2'}(a^2_{subj}) = \frac{5}{11} \times \frac{1}{3} + \frac{6}{11} \times \frac{1}{3} = \frac{1}{3} \nonumber \\
\text{Bayesian updates} \qquad & p_1^2 = \frac{\frac{5}{11} \times \frac{1}{3}}{\frac{1}{3}} = \frac{5}{11} \tag{4.16} \\
& p_2^2 = \frac{\frac{6}{11} \times \frac{1}{3}}{\frac{1}{3}} = \frac{6}{11}\ . \tag{4.17}
\end{align}$$

The posterior $P^1_{H_{s_2}}$ is therefore $\{p_1 = \frac{5}{11}, p_2 = \frac{6}{11}\}$.

Since this example is small and symmetrical, it is easy to see that if our observer had chosen to block *I-2* first, the posterior at s_2 would be $P^2_{H_{s_2}} = \{p_1 = \frac{6}{11}, p_2 = \frac{5}{11}\}$. In other words, the goal posteriors would be reversed despite $P^1_{H_{s_2}}$ and $P^2_{H_{s_2}}$ sharing identical system state s_2. The differences in posteriors will diverge more in domains that are not symmetrical, or if we do not assume uniform priors.

The imbalance in the number of plans remaining to the subject in state s_1' makes its first action more informative. By contrast, when both goals share plan counts in state s_2', the subject's action to move down is not discriminative. This suggests an emergent strategy on the part of the observer agent: creating imbalance in the number of plans to different goals could actually lead to greater divergence in our posteriors. This in turn informs where we should focus interventions since efforts are concentrated on goals that are more likely according to our posterior. To describe this strategy intuitively, if we make one goal more difficult to achieve by a certain path, e.g. by blocking options along that path, we can infer that an agent that still follows that path is more likely to be pursuing other goals. This property has consequences for how we model the problem in algorithm design. We need a model that can distinguish the nodes of T_{e_ρ} that differ only in the goal posterior. In other words, we need to model a belief state over the subject's possible goals.

4.5 AGRD as a POMDP

Active Goal Recognition Design can be reframed as a Partially-Observable Markov Decision Process (POMDP). A POMDP is a tuple $\langle S, \Omega, A, T, R \rangle$ where S is the state space, Ω is the space of possible observations of states $s \in S$, A is the set of available actions, T is a transition function that returns the probability of transitioning to state s' after taking action a in state s, and R is the reward function the agent receives for transitioning to a given state. We will formulate the POMDP as a Belief MDP where our belief state is composed of the system state as defined in an AGRD problem instance plus our belief about the subject's goal, which is fully summarized by the goal posteriors incrementally computed by our Bayesian update (Equation 4.6, Page 20).

We first examine the state space S and observations Ω. A state $s \in S = \langle \omega, g \rangle$ where ω is a fully observable state variable and g, the subject's true goal, is a hidden state variable. $\omega \in \Omega$ corresponds exactly to a state in our AGRD problem formulation: it describes the subject, observer, and their shared environment. Since the subject's goal is hidden, an observation could be consistent with multiple states. For example, if there are 3 possible goals in an AGRD problem instance, ω is consistent with $s_1 = \langle \omega, g_1 \rangle, s_2 = \langle \omega, g_2 \rangle$, and $s_3 = \langle \omega, g_3 \rangle$. However, a transition in an MDP can only result in a single successor state, so to reason about our POMDP, we formulate it as a Belief MDP where our state representation is a "belief" state that describes our belief about the current state we are in. This belief must be a probability distribution over possible current states and must be updated after each observation we receive, thereby fully summarizing the history of execution within the MDP. We have already defined our goal posteriors as incremental Bayesian updates given sequential observations, so let the belief state $b = \langle \omega, P^i_{H_\omega} \rangle$ where $P^i_{H_\omega}$ represents the ith possible goal posterior we could obtain at the observable state ω. We will need the posterior when describing how the POMDP transition function changes to operate on belief states.

The actions A of the POMDP include only the observer interventions, not subject actions. Executing an intervention a_{obs} on a state s alters some observable state deterministically. Let us call this intermediary state $\hat{s}$. The successor we ultimately transition to, s', is distributed by $T(s, a_{obs}, s')$, which models the uncertainty we have about which action a subject will take after a_{obs} is executed. Even if we know its goal, we still do not know which plan the subject is executing, so we define T in terms of the action likelihoods from Equation 4.4 (Page 20):

$$T(s, a_{obs}, s') = P_{\hat{s}}(a_{subj}|g) \tag{4.18}$$

Where

$$a_{obs}(s) = \hat{s}$$

$$a_{subj}(\hat{s}) = s'$$

$$s = \langle \omega, g \rangle, s' = \langle \omega', g \rangle \ .$$

However, the true goal g is hidden, and therefore the transition function T is also hidden as it requires exact knowledge of g. We therefore require T_b, a transition function that can operate on our belief states.

T_b distributes successors not by the likelihood of an action given a goal, but instead by the marginal likelihood from Equation 4.5 (Page 20):

$$T_b(b, a_{obs}, b') = P_{\hat{b}}(a_{subj}) \tag{4.19}$$

Where

$$a_{obs}(b) = \hat{b}$$

$$a_{subj}(\hat{b}) = b'$$

$$b = \langle \omega, P^i_{H_\omega} \rangle, b' = \langle \omega', P^i_{H_{\omega'}} \rangle \ .$$

The marginal likelihood is computed from priors over goals, here given as $P^i_{H_\omega}$. The posterior

$P^i_{H_{\omega'}}$ is computed by our Bayesian update (Equation 4.6), and represents the update of our belief state when a new observation is received.

In the AGRD problem, the objective is to minimize e_ρ. Therefore, the reward R for a state is simply $-\rho(s)$. Any belief state where the hypothesis $|H_\omega| \leq 1$ is defined as an absorbing state. Recall that in both Equations 4.7 and 4.8, $\rho(s) = 0$ if $|H_s| \neq 1$, so maximizing our reward is the same as minimizing the ρ value achieved in an absorbing state.

4.5.1 Solving AGRD

Having mapped our problem to an MDP, we briefly consider MDP solving algorithms.

Value Iteration [Bellman, 1957] and its many variants can optimally solve MDPs, however they require the state space upfront to perform Bellman Backups and converge to the optimal policy. This is problematic for our formulation since the state space could be very large, and to produce it upfront one needs to compute explicit transition functions and goal posteriors for every single state.

MDP solving algorithms based on RTDP are better suited to the tree structure of our domain. LRTDP [Bonet and Geffner, 2003] can converge to the optimal policy more quickly than Value Iteration. Since it explores the state space via "rollouts," state information can be lazily computed as it simulates transitions, and the exploration is necessarily limited to the reachable state space. However this is still inefficient as an optimal solution since convergence requires LRTDP to visit states with more than one branch multiple times in subsequent rollouts before all states are "labeled."

We therefore propose in the next chapter a specialized optimal algorithm for solving an Active Goal Recognition Design problem instance. Theorem 1 proves that T_{e_ρ}, the tree explored by the optimal solution, is a Directed Acyclic Graph. Therefore, unlike LRTDP and Value Iteration, each state of the augmented state space need only be visited at most once. Our algorithm will exploit this as it exhaustively explores every branch of the tree in a Depth-First Search.

CHAPTER 5

Optimal Algorithm

We now propose an algorithm to solve an AGRD problem instance optimally. We describe the algorithm in detail and present analysis to prove correctness. We evaluate its performance on a new set of benchmark domains for this problem, and finally we suggest enhancements.

5.1 Algorithm

Algorithm 1: *Opt-AGRD*: Optimal AGRD

```
   Input  : $I$ - problem instance
   Output: Achieved $\rho$ once the goal is known.
1  let $D, s_{root}, \mathcal{G}, P_{\mathcal{G}}, \rho$ refer to the components of $I$
   // Initialize Goal Hypothesis
2  $H \leftarrow$ GoalHyp($\mathcal{G}, s_{root}, s_{root}$)
3  $P_H \leftarrow P_{\mathcal{G}}$
4  $s \leftarrow s_{root}$
5  while $|H| > 1$ do
6      $\_, a_{min} \leftarrow$ DFS-Intervention($D, \rho, H, P_H, s$)
7      $s \leftarrow a_{min}(s)$
8      $s' \leftarrow$ ObserveSubjectTransition($s$)
9      $H, P_H \leftarrow$ UpdateHypothesis($H, P_H, s, s'$)
10     $s \leftarrow s'$
11 return $\rho(s)$
```

Opt-AGRD is a straightforward encoding of the optimal solution presented in Section 4.3. We initialize the goal hypothesis and the probability of each goal in the hypothesis to the set of all goals and their priors [Lines 2-3]. The main loop invokes DFS-Intervention, which is mutually recursive with DFS-Action, to determine which intervention produces the best score for the current state. It executes the best intervention, then observes the subject's action. It computes the new goal hypothesis and posterior goal probabilities by taking the goal priors and conditioning them (with normalization) on the subject's observed transition [Lines 7-9]. These posteriors are used as the priors for the next iteration.

Proposition 1 *Opt-AGRD implements Equation 4.3 (Page 19) and, given a state and goal priors, applies the optimal intervention to the domain.*

Proof: As long as DFS-Intervention returns the optimal intervention (Proposition 2), *Opt-AGRD* applies the optimal intervention to the domain [Line 7]. □

DFS-Intervention and DFS-Action, Algorithms 2 and 3 respectively, explore T_{e_ρ} depth-first to produce a score for each visited node. DFS-Intervention takes the minimum over observer interventions, and DFS-Action takes the expectation over subject actions. The termination condition of the DFS tree is when $|H| \leq 1$ [Line 12], corresponding to when the subject's goal is uniquely distinct. If not a leaf, additional child nodes are generated by simulating all possible actions and descending down the tree depicted in Figure 4.1 [Lines 16-17, 25-26]. Section 4.5 introduced belief states $b = \langle \omega, P^i_{H_\omega} \rangle$ encompassing both the system state and the goal posteriors. Accordingly, DFS-Intervention and DFS-Action track these posteriors through the tree and update them after each simulated subject action [Line 27].

DFS-Intervention simulates the observer interventions by trialing them all and keeping the minimum. Note that the interventions returned by $A_{obs}(s_{sim})$ are restricted to only those that do not change the value of $C^*(g)$ for all goals in the goal hypothesis H, therefore we do not update the goal hypothesis after simulating an intervention.

Proposition 2 DFS-Intervention *implements Equation 4.1 (Page 19) and returns both the optimal score and intervention for the given state and posteriors.*

Proof: If the state being evaluated is a leaf, DFS-Intervention returns immediately with that state's score and the Identity intervention [Line 13], which is the optimal intervention

Algorithm 2: DFS-Intervention:
Depth First Search for Intervention iteration

```
Input  : $D$ - Domain,
         $\rho$ - value function,
         $H$ - current goal Hypothesis,
         $P_H$ - goal priors for Hypothesis,
         $s_{sim}$ - simulated subject state
Output: score, intervention - $\mathbb{R}$ score value and the intervention that produced it
12 if $|H| \leq 1$ then
13     return $\rho(s_{sim})$, Identity
   // Simulate Observer interventions
14 score $\leftarrow \infty$
15 intervention $\leftarrow$ Identity
16 foreach $a_i \in D.A_{obs}(s_{sim})$ do
17     $s' \leftarrow a_i(s_{sim})$
18     score $\leftarrow$ min(score, DFS-Action($D, \rho, H, P_H, s'$))
19     if score changed then
20         intervention $\leftarrow a_i$
21 return score, intervention
```

at a leaf node, i.e. no action. Otherwise, DFS-Intervention returns the intervention and score corresponding to the minimum score of its successors, which is optimal if the scores returned by DFS-Action are optimal (Proposition 3). □

DFS-Action simulates the subject's action. Until we have reached a leaf, we are uncertain of the subject's goal. Additionally, if multiple actions could lead to the same goal, we are uncertain of which the subject would take. Therefore, we must take the expectation of the score over all possible actions the subject might take in a given state. DFS-Action obtains a list of actions applicable in the simulated state and then computes the probability of each action [Lines 22-23]. We discuss the ActionProb function in more detail in Algorithm 4, but for now observe that the output is a probability distribution over the available actions passed to the function. We take the expected score of the available actions over this distribution [Line 29]. Note that it is possible the probability of an action is 0. In this case, the action score will not change, and in practical implementations this action can be pruned from the search.

```
Algorithm 3: DFS-Action:
Depth First Search for Action iteration
  Input  : $D$ - Domain,
           $\rho$ - value function,
           $H$ - current goal Hypothesis,
           $P_H$ - goal priors for Hypothesis,
           $s_{sim}$ - simulated subject state
  Output: score - $\mathbb{R}$ score value
22 $subjectActions \leftarrow D.A_{subj}(s_{sim})$
23 $P_A \leftarrow$ ActionProb($actions, H, P_H, s_{sim}$)
24 score $\leftarrow 0$
25 for $a_i \in subjectActions$ do
26     $s' \leftarrow a_i(s_{sim})$
27     $H', P'_H \leftarrow$ UpdateHypothesis($H, P_H, s_{sim}, s'$)
28     $trial, \_ \leftarrow$ DFS-Intervention($D, \rho, H', P'_H, s'$)
29     $score \leftarrow score + (P_A[i] * trial)$
30 return score
```

Proposition 3 DFS-Action *implements Equation 4.2 (Page 19) and returns the optimal score for the given state.*

Proof: DFS-Action simulates every action and computes the updated hypothesis given that successor. It accumulates [Line 29] and eventually returns the expected score [Line 30] of its successors as computed by DFS-Intervention and weighted by P_A. As long as ActionProb returns the correct action probabilities (Proposition 5), DFS-Intervention is passed the accurate goal posteriors and goal hypothesis (Proposition 7), and DFS-Intervention returns the optimal score (Proposition 2), ActionProb returns the optimal score.

We note that the proofs of Proposition 2 and Proposition 3 depend on each other. However, this recursive proof has a "terminating condition" since DFS-Intervention will eventually return optimal scores without invoking DFS-Action. As long as this terminating condition holds (Proposition 4), the mutually dependent proofs hold. □

Proposition 4 *The recursion between* DFS-Intervention *and* DFS-Action *will eventually terminate with* DFS-Intervention *returning without invoking* DFS-Action.

Proof: Theorem 3 (Page 24) proves a finite limit on κ, the depth bound of T_{e_ρ} (the tree induced by e_ρ). Since DFS-Intervention and DFS-Action implement Equations 4.1 and 4.2 respectively, they explore the same tree. DFS-Intervention checks for the leaf node condition, i.e. $|H_s| \leq 1$, on Line 12 and returns if it is met. Therefore, since the tree has a finite depth bound and the leaf condition stops recursion, the recursion between DFS-Intervention and DFS-Action is guaranteed to terminate. □

Algorithm 4: ActionProb:
Computing Probability of Actions in a state

```
   Input  : $actions$,
            $\mathcal{G}$ - goal set,
            $P_{\mathcal{G}}$ - goal priors,
            $s_{sim}$ - simulated state
   Output: Sequence of probabilities for $actions$
31 init $P_A$ as array of length $|actions|$ to 0
32 foreach $a_i \in actions$ do
33     $s' \leftarrow a_i(s_{sim})$
34     $P' \leftarrow$ ConditionedPriors($\mathcal{G}, P_{\mathcal{G}}, s_{sim}, s'$)
35     $P_A[i] \leftarrow \sum_{p \in P'} p$
36 return $P_A$
```

DFS-Action requires a probability distribution for the actions available in a state. This distribution must reflect that actions leading to goals that are more likely (as expressed by our priors P_H) have a higher probability and our assumption that all plans to a given goal are equally likely. Algorithm 4 defines ActionProb, which computes this distribution. It does this by considering which goals the subject could be pursuing and how likely it is that a subject pursuing any of those goals would take each of the actions in our action list. From an action, we obtain the successor state s' and compute the probability of all goals conditioned on the subject transitioning from s_{sim} to s'. These conditioned probabilities are not normalized, and thus do not sum to 1. Instead, the sum of these priors for an action over all goals comes to the probability that a subject would take the action [Lines 32-35]. Note that if s' is not on the optimal path to some goal g_j, the conditioned probability of that goal will be 0 since an optimal subject pursuing that goal would not take that action. The sum of probabilities in P_A will be 1 because in summing all conditioned goal probabilities for all actions, we account for all possibilities.

Proposition 5 ActionProb *implements the sum operation of Equation 4.5 (Page 20) and returns the marginal likelihood of every action passed to it.*

Proof: For every action, ActionProb sums the goal priors conditioned on the likelihood of the action given each goal. As long as ConditionedPriors returns the prior times likelihood for the given transition (Proposition 6), ActionProb returns the marginal likelihood of all actions. □

```
Algorithm 5: ConditionedPriors:
Get probabilities for goals conditioned on the subject's action
  Input  : $\mathcal{G}$ - goal set,
           $P_{\mathcal{G}}$ - goal priors,
           $s$ - simulated or current state,
           $s'$ - successor state
  Output: Sequence of goal priors conditioned on the subject transitioning to $s'$
37 init $P'$ of size $|P_{\mathcal{G}}|$ with all elements equal to 0
38 foreach $g_j \in \mathcal{G}$ do
       // Probability of goal j
39     let $p_j \in P_{\mathcal{G}}$
40     $P'[j] \leftarrow p_j \frac{\text{PlansToGoal}(g_j, s')}{\text{PlansToGoal}(g_j, s)}$
41 return $P'$
```

Algorithm 5 defines ConditionedPriors. This function returns a set of goal probabilities from priors conditioned on the likelihood of the subject transitioning from s to successor s'. For each goal, we multiply the existing goal prior by the fraction of optimal plans to that goal that pass from s to s' [Lines 38-40]. This fraction captures the fact that we do not know which plan the subject is pursuing, so we assume that all plans are equally likely.

Proposition 6 ConditionedPriors *implements Equation 4.4 (Page 20) and the likelihood times prior portion of Equation 4.5. It returns the unnormalized posteriors over the goal set given the transition to successor s'.*

Proof: This trivially follows from the algorithm's definition. □

One could imagine various ways of ranking plans to make some more likely than others, e.g. in an adversarial setting, though the exploration of more complex methods is beyond the scope of this thesis.

Algorithm 6: UpdateHypothesis:
Updates Goal Hypothesis and Posterior given a transition

```
   Input  : $\mathcal{G}$ - goal set,
            $P_{\mathcal{G}}$ - goal priors,
            $s$ - simulated or current state,
            $s'$ - successor state
   Output: New Goal Hypothesis and Posterior conditioned on the given transition
42 $H \leftarrow$ GoalHyp($\mathcal{G}, s, s'$)
43 $P'_H \leftarrow$ ConditionedPriors($H, P_{\mathcal{G}}, s, s'$)
44 $P_H \leftarrow$ Normalize($P'_H$)
45 return $H, P_H$
```

Algorithm 6 defines the function UpdateHypothesis. It takes a set of goals and their priors as well as a transition from s to s' and outputs a new hypothesis and set of posterior probabilities. UpdateHypothesis is invoked both to update the hypothesis after we receive an observation [Line 9] and in exploring the state space when simulating subject actions [Line 27]. Note that when a transition diverges from all optimal paths to some goal, that goal is pruned from H.

Proposition 7 UpdateHypothesis *implements Equation 4.6 (Page 20) and returns the correct goal posteriors given a transition along with the updated goal hypothesis.*

Proof: GoalHyp merely prunes goals from the hypothesis that cannot be optimally reached via the transition. Normalize takes an input array and normalizes its values. Therefore, as long as ConditionedPriors returns an array of the goal priors times the likelihood of the transition (Proposition 6), UpdateHypothesis returns the correct posteriors and goal hypothesis for the transition. □

Algorithm 7: GoalHyp:
Get an optimal subject's Goal Hypothesis at a state

```
Input  : $\mathcal{G}$ - possible goal set,
         $s$ - simulated or current state,
         $s'$ - successor state
Output: Set $\mathcal{G}' \subseteq \mathcal{G}$ an optimal subject could be pursuing
46 let $\Phi^* = \{$all possible $\phi^*_{s,g}\ \forall g \in \mathcal{G}\}$
47 let $\Phi \subseteq \Phi^*$ be all optimal subject plans through $s'$
48 init $\mathcal{G}' = \emptyset$
49 add $g$ to $\mathcal{G}'$ for all $\phi_{s,g} \in \Phi$
50 return $\mathcal{G}'$
```

Algorithm 8: PlansToGoal:
Enumerate all optimal plans from a state to a goal

```
Input  : $g, s$
Output: $\mathbb{Z}^{\geq}$ - number of plans
51 let $\Phi^* = \{$all possible $\phi^*_{s,g}\}$
52 return $|\Phi^*|$
```

Algorithm 7, GoalHyp, reports all goals from the given set $\mathcal{G}$ for which there exists any optimal subject plan from s that passes through the state s'. Algorithm 8, PlansToGoal, takes a goal g and a state s and returns the number of optimal subject plans that exist between s and g.

Theorem 5 *Opt-AGRD always executes the optimal intervention.*

Proof: This theorem differs from Proposition 1 in that it is a stronger statement. Propositions 1-7 prove that the algorithms faithfully implement the equations defining an optimal solution in Section 4.3. For this theorem to hold true, the goal posteriors must be correctly updated after receiving every observation. This occurs in Line 9, so the theorem holds. □

5.1.1 Computing the Count of Plans to Goals

Opt-AGRD requires us to be able to query the number of optimal subject plans to each goal from any state. In practice, this is accomplished by performing a Dijkstra-like search of the

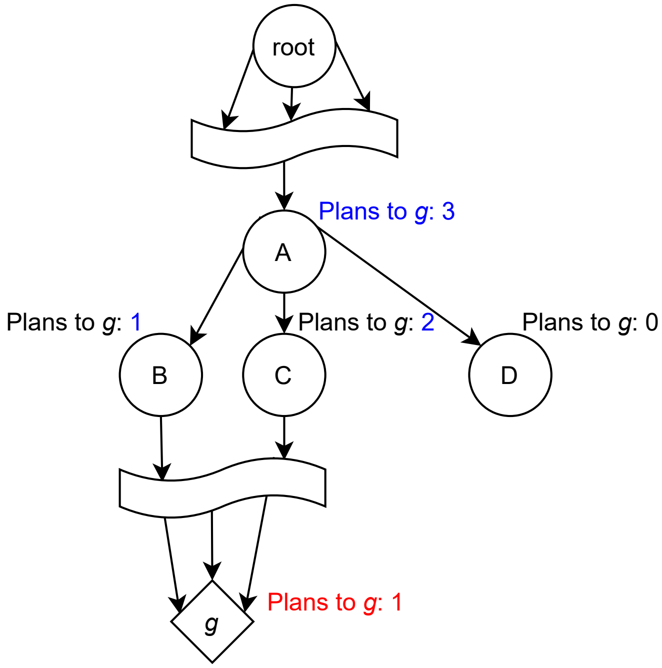

Figure 5.1: Walking back the number of plans to goal g from each state via parent pointers

state space rooted in the current state where successors are defined exclusively by subject actions. The minimum cost path to reach each goal is recorded, and the search terminates once all paths to all goals of cost equal to that goal's minimum cost have been discovered. For domains with a heuristic function available, our implementation utilizes a simple heuristic for a multi-goal search: given a state, use the minimum heuristic value out of all goals.

Once the forward search finishes, we cache the number of plans to each goal from every state by walking back from the goals. We seed the open list with the discovered goals, then visit ancestor nodes breadth-first via parent-pointers. For a given goal and a given state, the number of plans from that state to the goal is the sum of the number of plans to that goal cached on all successors. See Figure 5.1 for an illustration. Let node A be a node expanded in this back-walk. A has successors B, C, and D, each of which have cached counts of 1, 2, and 0 plans to goal g respectively. To determine how many plans are available to g from node A, we sum the counts cached with all successors: $1+2+0=3$. When goals are seeded into the open list, they are initialized with a plan count of 1.

This information must be recomputed frequently as the algorithm descends down the search tree depth-first. Conceptually, every intervention executed on a state s (besides *Identity*) transitions to a new state s', thus invalidating all previous subject plans $\phi^*_{s,g}$.

It is conceivable to design a domain where the subject's state can be isolated within the system state so that subject plans before and after an intervention stay mostly the same. In the domains we evaluate on, there is significant overlap between plans available from s and s'. In other words, the states are slightly modified, but the actions available to the subject in those states are mostly identical. Even so, in the general case, plan counts cached with state variables common to both s and s' cannot be trusted after an intervention is executed since arbitrary actions could be invalidated. Even if we have a domain where we are able to immediately identify an action that has been invalidated and the shared state whose cache is directly affected, we would still have to propagate back the updated plan count discounting that action to all ancestors of the state. Additionally, the observer is permitted to add actions to the domain as long as they don't change the costs to any goal. This requires us to re-search for subject plans from the current state to ensure we accurately count all *new* plans enabled for the subject.

5.2 Evaluation

To characterize the behavior of *Opt-AGRD*, we implemented it in C++ and ran experiments on Intel E8500 3.16 GHz CPUs. Our primary concern for this evaluation is how our implementation scales, i.e., which domain characteristics affect runtime the most. As a testbed for our experiments, we use grid pathfinding with 4-way movement and a hand-coded version of the Logistics domain.

For our grids, we used two patterns of obstacles. In uniform grids, each cell had a 0.2 probability of being blocked. Room grids contain walls with randomly placed openings, scaled down from those in the Moving AI repository [Sturtevant, 2012]. In both domains, the subject and observer's start locations and possible goals were chosen uniformly at random from among unblocked cells. The remaining unblocked cells had a 0.3 chance of being

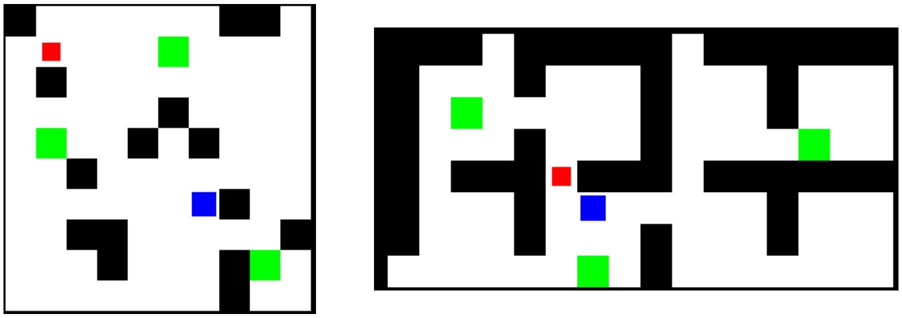

Figure 5.2: Examples of Uniform (left) and Rooms (right) Gridworld domains. Black cells are obstacles, green are goals, red is the subject, and blue is the observer.

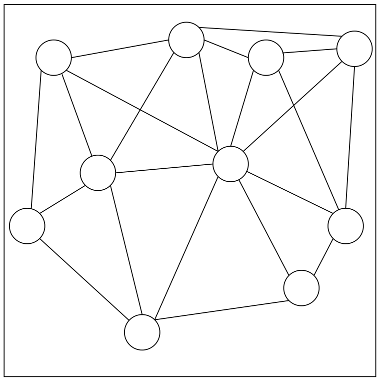

Figure 5.3: Example of network topology used for Logistics domain.

blockable by observer interventions. The observer follows the same movement rules as the subject and both can occupy the same cell. To block a cell, the observer must be adjacent to it and it must be marked as blockable. This is nearly identical to the terrorist domain used in previous work [Pozanco *et al.*, 2018; Shvo and McIlraith, 2020]. See Figure 5.2 for examples.

In Logistics, the object is to move packages to specific locations matching a goal specification. Packages must be moved by trucks that can drive between locations. Locations and the network connecting them are defined by sampling points in a unit square using a connection distance of 0.4. The observer follows the same movement rules as the subject trucks and can remove connections to other locations from the location they currently occupy. All goals generated for our experiments specify a location for 1 package. The goal package and location were chosen uniformly at random. See Figure 5.3 for an example of the network of locations.

Instances for all domain types were generated with 2, 3, or 4 goals. We only report on instances where observer interventions can affect $e_\rho(s_{root})$, which filtered out many instances. Results are reported on 531 Uniform Grids, 44 Rooms Grids, and 1120 Logistics instances.

Several factors clearly will not affect the running time of the algorithm:

1. *The length of the plan the subject executes to achieve its goal:* Once the subject diverges from all plans to other goals, we have achieved goal recognition. Therefore, the actual length of the subject's plan to its goal is irrelevant.

2. *The prior distribution over goals: Opt-AGRD* must examine all possible branches of T_{e_ρ} to ensure optimality, no matter how unlikely a goal may be.

3. *The size of the state space:* Assuming the optimal plan lengths to goals are held constant, the explored states are limited to those required to find plans to goals.

We did examine runtime as a function of the number of goals, which affects the branching factor of T_{e_ρ} due to the actions a subject could take to additional goals. This did appear to play a minor role in runtime, but was easily dominated by the depth of the tree. We did not examine runtime as a function of the number of interventions, as this should have a similar effect as the number of goals, namely an increased branching factor.

The dominating factor in runtime was the length of the optimal plan to the second-furthest goal. This follows from Theorem 3, which establishes the depth bound of T_{e_ρ} based on the second-furthest goal. Figure 5.4 plots the geometric mean of the runtime to exhaustively explore T_{e_ρ} with 95% confidence intervals as a function of this upper bound. (No room maps with an upper bound of 2 survived filtering.) To obtain this runtime, we only report on the algorithm's time to return the first intervention, which necessarily requires it to expand the entire tree exactly one time. The plot shows that, as one might expect, *Opt-AGRD* scales exponentially in the depth of the tree. The domain instances used in our experiments had varying numbers of cells / locations, but the overwhelming factor was the depth of the tree.

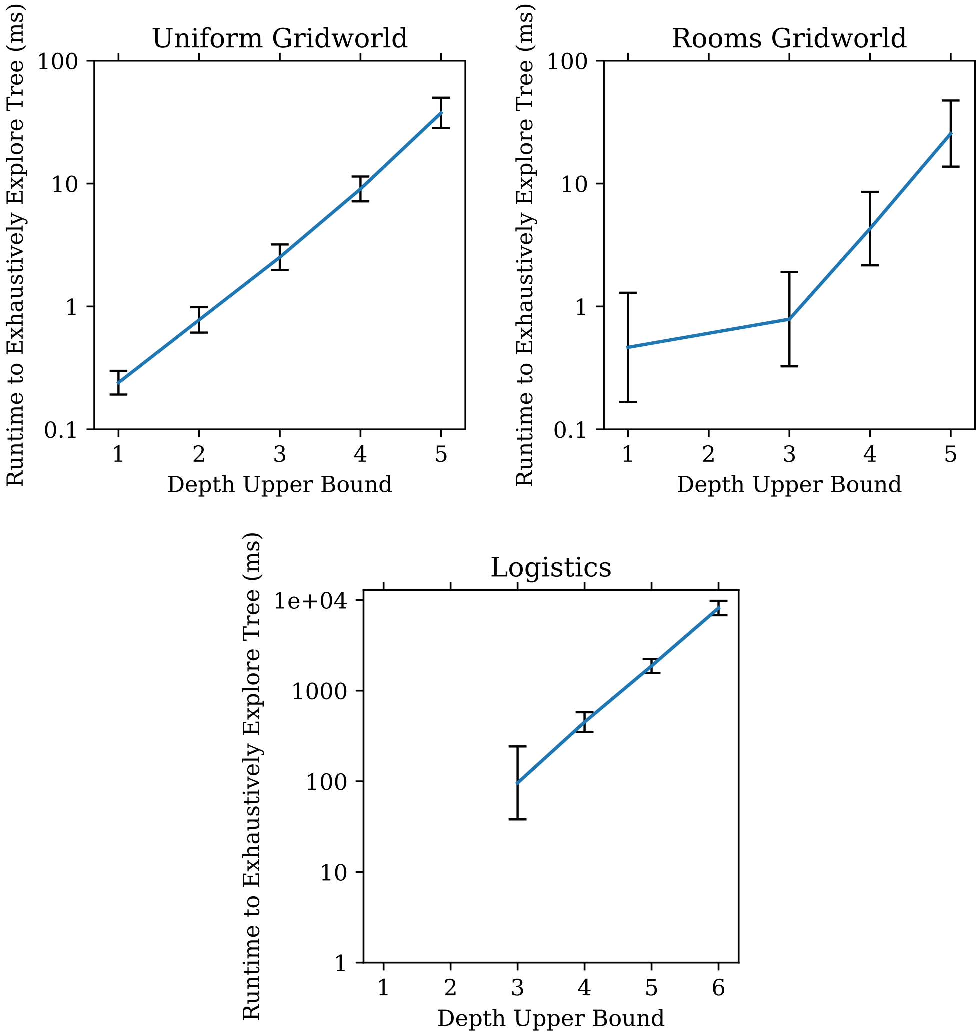

Figure 5.4: Avg runtime to explore the tree as a function of search tree depth.

5.3 Enhancements

In domains where the observer is required to move in some physical space or network (e.g. GridWorld and Logistics), many of the interventions are simply movement, and thus are guaranteed to affect no subject plans. In all domains, the *Identity* intervention is explicitly defined as one that affects no state, so by definition does not affect subject plans. In our implementation, we detect these interventions and avoid recomputing subject plan counts when simulating them in our tree search.

The remaining enhancements discussed were not implemented for our experiments. We present them here as possible directions for future work.

Dynamic Programming We proved in Theorem 4 that T_{e_ϱ} exhibits optimal substructure, which is a key requirement for using dynamic programming to speed up computation by caching optimal solutions to sub-problems. We then argued that this optimal substructure did not overlap as much as it first appeared, undercutting the case for dynamic programming. Nevertheless, we point to a domain like GridWorld where many of the observer's interventions are simply movement. Consider these two sequences of observer interventions (ignoring the subject actions that would be interleaved): $\langle LEFT, RIGHT, LEFT\rangle$ and $\langle RIGHT, LEFT, RIGHT\rangle$. Assuming the subject takes the same actions for either sequence of interventions, it is clear that the system state and goal posteriors will be identical. We therefore propose that dynamic programming solutions could provide speedups in some domains, specifically those where much of the time the observer is likely to execute interventions that do not change state relevant to the subject (or subject plans).

Tree Pruning The algorithm we have presented exhaustively explores every branch of the tree, but this is not required to achieve optimality. We can adapt the famous Alpha-Beta Pruning algorithm [Knuth and Moore, 1975] to our expectimin tree. Consider the root node of the tree. We wish to determine the intervention that leads to the minimal score. To find the score for an intervention, we compute the expected score over all children of that intervention, i.e. subject actions. This expected value is a sum accumulated incrementally by evaluating each subject action sequentially and adding the result, weighted by the probability of the action. Consider a case where the algorithm has computed a score for one intervention at the root. As the only score computed, it is by default the minimal score *so far*. Let us call this intervention the incumbent. The algorithm then computes the score for the next intervention by accumulating the expected score of its children. If the accumulated score over some subset of all actions exceeds the incumbent's score, we need not evaluate the remaining actions since we know the incumbent still has the minimal score. This optimization also works at any subtree rooted in the successor of a subject action, so can be employed recursively.

While tree pruning is not guaranteed to provide any speedup, it has the potential to prune a lot of branches depending on the variance between branch scores and the order they are evaluated. One could employ domain-specific heuristics informing the search on which subject actions should be explored first to maximize branch pruning.

5.3.1 Algorithmic Framework

The algorithmic framework presented as Algorithms 1, 2, and 3 can be parameterized to facilitate a differing set of assumptions. The functions ConditionedPriors and GoalHyp (Algorithms 5 and 7 respectively) compute their values based on optimal subject plans, but modeling different assumptions could be accomplished through refinement or redefinition of these methods. For instance, adversarial subjects could be modeled by changing the calculation of the likelihood of an action in a simulated state to make ambiguous actions more likely and discriminating actions less likely. Sub-optimal subjects could be modeled by redefining the likelihood using all plans bounded by some sub-optimality measure rather than all optimal plans.

We believe the flexibility of this framework can support a wide range of assumptions beyond those stated thus far. In Chapter 6, we will use this framework to formulate a suboptimal algorithm that differs primarily in how the action likelihoods are computed (Algorithm 4). A much richer exploration of differing assumptions within this framework is a promising research direction, and we leave these extensions to future work.

CHAPTER 6

Suboptimal Algorithm

In the previous chapter we presented an algorithm that solves an AGRD problem instance optimally. We also noted in our evaluation that the runtime of the algorithm scales exponentially in the depth of the tree explored. However, the AGRD problem setting is designed for online applications where exponential runtime is likely infeasible. We therefore explore a simple anytime algorithm based on *Opt-AGRD* that finds sub-optimal solutions quickly and is able to improve solutions over time.

6.1 Iterative Widening: *IW-AGRD*

We retain the same assumptions and restrictions, namely that the subject acts optimally, all subject plans to a given goal are equally likely, and the observer cannot change the optimal cost to any subject goal that has not been pruned yet.

The key idea is to reduce the number of branches the algorithm explores by aggressively pruning subject actions from consideration. The algorithm is given a time bound for how long each search episode is allowed to take to find the best intervention for the current state. At the start of each episode, we set our pruning parameter to eliminate all but the most likely actions and exhaustively explore this significantly reduced tree. At the end of each of these iterations, we cache the chosen intervention. We then relax our pruning parameter slightly and perform another iteration exploring the widened tree, reminiscent of Ginsberg and Harvey's Iterative Broadening [1992]. Once the episode's time bound is reached, we return the cached intervention, which corresponds to the intervention selected by the most recently finished search iteration.

To implement this idea, we use the structure of *Opt-AGRD*, but we replace Algorithm 4 with a modified calculation of action likelihood.

Algorithm 9: Iterative Widening ActionProb: Action Probability with pruning

```
   Input  : $actions$,
            $\mathcal{G}$ - goal set,
            $P_{\mathcal{G}}$ - goal priors,
            $s_{sim}$ - simulated state,
            $w$ - Integer that controls "width" of the tree
   Output: Sequence of probabilities for $actions$
   // Compute likelihoods as before
53 init $P_A$ as array of length $|actions|$ to 0
54 foreach $a_i \in actions$ do
55     $s' \leftarrow a_i(s_{sim})$
56     $P' \leftarrow$ ConditionedPriors($\mathcal{G}, P_{\mathcal{G}}, s_{sim}, s'$)
57     $P_A[i] \leftarrow \sum_{p \in P'} p$
   // Keep only the most likely actions
58 $topWLikelihoods \leftarrow$ TopNUnique($w, P_A$)
59 $threshold \leftarrow min(topWLikelihoods)$
60 foreach $p_i \in P_A$ do
61     if $p_i < threshold$ then
62         $p_i \leftarrow 0$
63 $P_A \leftarrow$ Normalize($P_A$)
64 return $P_A$
```

Lines 53-57 are identical to Algorithm 4, and changes are colored blue. Starting at Line 58, we prune actions by setting an action's probability to 0 if it does not meet a threshold. This threshold is determined by getting the top N unique likelihood values from P_A where $N = w$, our integer widening parameter. The minimum value of this group of top N likelihoods is set as the threshold. Finally, we normalize the pruned action probabilities [Line 63] so that our action probabilities still sum to 1. In effect, we discard any actions that do not meet the threshold by returning a distribution that excludes them.

This altered version of ActionProb is invoked within DFS-Action (Alg. 3) in place of Alg. 4. DFS-Action therefore must accept the widening parameter w, but is otherwise unchanged.

To execute the tree search iteratively, we must also alter the top-level process of *Opt-AGRD*.

Algorithm 10: *IW-AGRD*: Iterative Widening AGRD

```
Input  : $I$ - problem instance,
         $timeBound$ - time bound for each search episode
Output: Achieved $\rho$ once the goal is known.
65 let $D, s_{root}, \mathcal{G}, P_{\mathcal{G}}, \rho$ refer to the components of $I$
   // Initialize Goal Hypothesis
66 $H \leftarrow$ GoalHyp($\mathcal{G}, s_{root}, s_{root}$)
67 $P_H \leftarrow P_{\mathcal{G}}$
68 $s \leftarrow s_{root}$
69 while $|H| > 1$ do
70     $w \leftarrow 1$
71     $a_{min} \leftarrow$ Identity
72     while $timeBound$ not reached do
73         $\_, a_{min} \leftarrow$ DFS-Intervention($D, \rho, H, P_H, s, w$)
74         $w \leftarrow w + 1$
75     $s \leftarrow a_{min}(s)$
76     $s' \leftarrow$ ObserveSubjectTransition($s$)
77     $H, P_H \leftarrow$ UpdateHypothesis($H, P_H, s, s'$)
78     $s \leftarrow s'$
79 return $\rho(s)$
```

There are two main differences between Algorithm 10 and Algorithm 1. First, we take *timeBound* as an additional parameter. We use this bound to control the loop executing the iterations [Line 72]. Once time has run out, we break from the loop and apply the most recently returned intervention. For simplicity, the pseudocode does not depict an "interrupt" mechanism, but note that it is possible for time to run out before an iteration has completed. If no iterations have completed before time runs out, the algorithm applies the Identity intervention initialized on Line 71.

The second alteration is the use and maintenance of the width parameter w. At the start of each search episode, we initialize w to 1 [Line 70], meaning only the most likely actions will be explored. DFS-Intervention is trivially modified to accept w as a parameter. It passes w along to DFS-Action, and vice versa, when control is traded between them. w is passed unaltered through the tree and is used in every invocation of Algorithm 9. Once a search iteration ends, w is incremented [Line 74] so that the next iteration, the likelihood threshold will be lower, so more actions will be considered.

We specifically note that the UpdateHypothesis function (Algorithm 6) is unaltered in *IW-AGRD*. Despite pruning many actions from our search, we still incorporate those actions in our Bayesian update (Equation 4.6). The intention with *IW-AGRD* is to reduce the exploration of the tree, not introduce error into the posteriors.

Since actions with probability 0 do not have to be expanded, the effect of the changes made to ActionProb in Algorithm 9 is that DFS-Action explores fewer branches of the tree. The savings in exploration of course come at the cost of optimality since we can no longer claim to have exhaustively explored the tree, but a focus on the most likely actions the subject could take should produce a good approximation of the value of an intervention.

6.2 Analysis

We now analyze *IW-AGRD* as compared to *Opt-AGRD*. Recall that T_{e_ρ} is the tree induced by e_ρ, and that *Opt-AGRD* exhaustively explores this tree to produce the optimal intervention. Theorem 2 proves that the size of T_{e_ρ} is finite under the reasonable assumption that there is an upper bound on the number of actions available to the observer and the subject in any given state. We will maintain that assumption for this analysis.

Let the specific branching factor for subject actions be defined as follows:

$$b_{subj} = \max_{s \in S} |A_{subj}(s)| \; . \tag{6.1}$$

Let us also define T^i_{IW} as the tree explored by *IW-AGRD* on iteration i of a search episode. The main difference between T_{IW} and T_{e_ρ} is that not every branch representing a subject action is necessarily included in T_{IW}.

Theorem 6 *Given a large enough time bound, IW-AGRD will eventually converge to the optimal intervention.*

Proof: We first note that for $T_{IW}^{i}, i = w$ since w is incremented after every iteration. Note also that $T_{IW}^{i} = T_{e_\rho}$ when w is large enough that every branch of the tree will be explored. Let us characterize when this happens. w is a lower bound on the number of actions that can be explored at every subject decision node of T_{IW}^{i}. This follows from the fact that we keep every action whose likelihood is equal to or greater than the wth-highest likelihood. When $w \geq b_{subj}$, we are therefore guaranteed that all subject actions will be explored, i.e. $T_{IW}^{i \geq b_{subj}} = T_{e_\rho}$.

Theorem 2 states that T_{e_ρ} is finite. Since *IW-AGRD* only prunes actions, $T_{IW} \subseteq T_{e_\rho}$, and therefore T_{IW}^{i} is finite for all values of i. Let x refer to the time it takes *Opt-AGRD* to explore T_{e_ρ} exhaustively. We can upper-bound the time it will take for *IW-AGRD* to converge to the optimal solution as $O(b_{subj}x)$. In other words, the time *IW-AGRD* will take to converge can be at most the time it takes to explore T_{e_ρ} a total of b_{subj} times in the situation where all subject actions have different probabilities. □

Although our suboptimal strategy reduces the branching factor of the tree by pruning actions, all trees T_{IW} inherit the same depth bound κ from T_{e_ρ} via Theorem 3.

Lemma 3 *For all i, the depth of T_{IW}^{i} is bounded by the second most expensive goal $\ddot{g} \in \mathcal{G}$.*

Proof: This follows directly from Theorem 3 which holds because of two properties that T_{IW} shares with T_{e_ρ}:

1. Leaves are defined as nodes where the subject's goal is distinct.

2. The observer is not allowed to change the optimal cost to any goal.

Property 1 means that the proof from Theorem 3 holds assuming goal costs remain static and Property 2 guarantees this fact. □

6.3 Evaluation

We evaluated the performance of *IW-AGRD* against *Opt-AGRD* using the same experimental domains as our runtime evaluation in Section 5.2.

Since every domain instance has multiple goals the subject could choose, we ran each domain instance with every possible actual subject goal. To match our assumptions, the subject's plan is chosen uniformly at random from among all possible optimal plans to its goal. When an intervention alters any state variables that affect the subject, the subject replans and again chooses randomly among all remaining plans to its goal. In our domains, a state variable that "affects the subject" is any state variable besides the observer's physical location in the system. To reduce variance in our results, we tested each instance and goal combination with 10 different random seeds. All experiment configurations were tested with *IW-AGRD* using time bounds of 1ms, 10ms, 100ms, and 1000ms.

The primary quality metric used for our experiments is ψ, the fraction of the subject's actual path that was non-distinct. To compare with the optimal solution, we establish an optimal baseline. This baseline was computed by averaging the ψ value achieved by *Opt-AGRD* over all seeds for each unique combination of domain instance and actual subject goal. For each experiment with *IW-AGRD*, we take the signed difference between this baseline and the actual achieved ψ, then divide it by the baseline. The result is the amount that the suboptimal solution deviated from the optimal solution, expressed as a signed percentage of the optimal solution's quality. For instance, 5% indicates that *IW-AGRD* was 5% worse than optimal, whereas -5% would indicate better performance than optimal. Note that due to the stochasticity of the domain, it is possible for a suboptimal algorithm to perform better than optimal in specific instances, though the optimal algorithm should do better on average.

We excluded from our results any experiments where both the baseline and the achieved ψ was 1.0, which corresponds to experiments where the subject's goal was not distinguished until it was achieved both by *Opt-AGRD* and *IW-AGRD*. In these experiments, there were

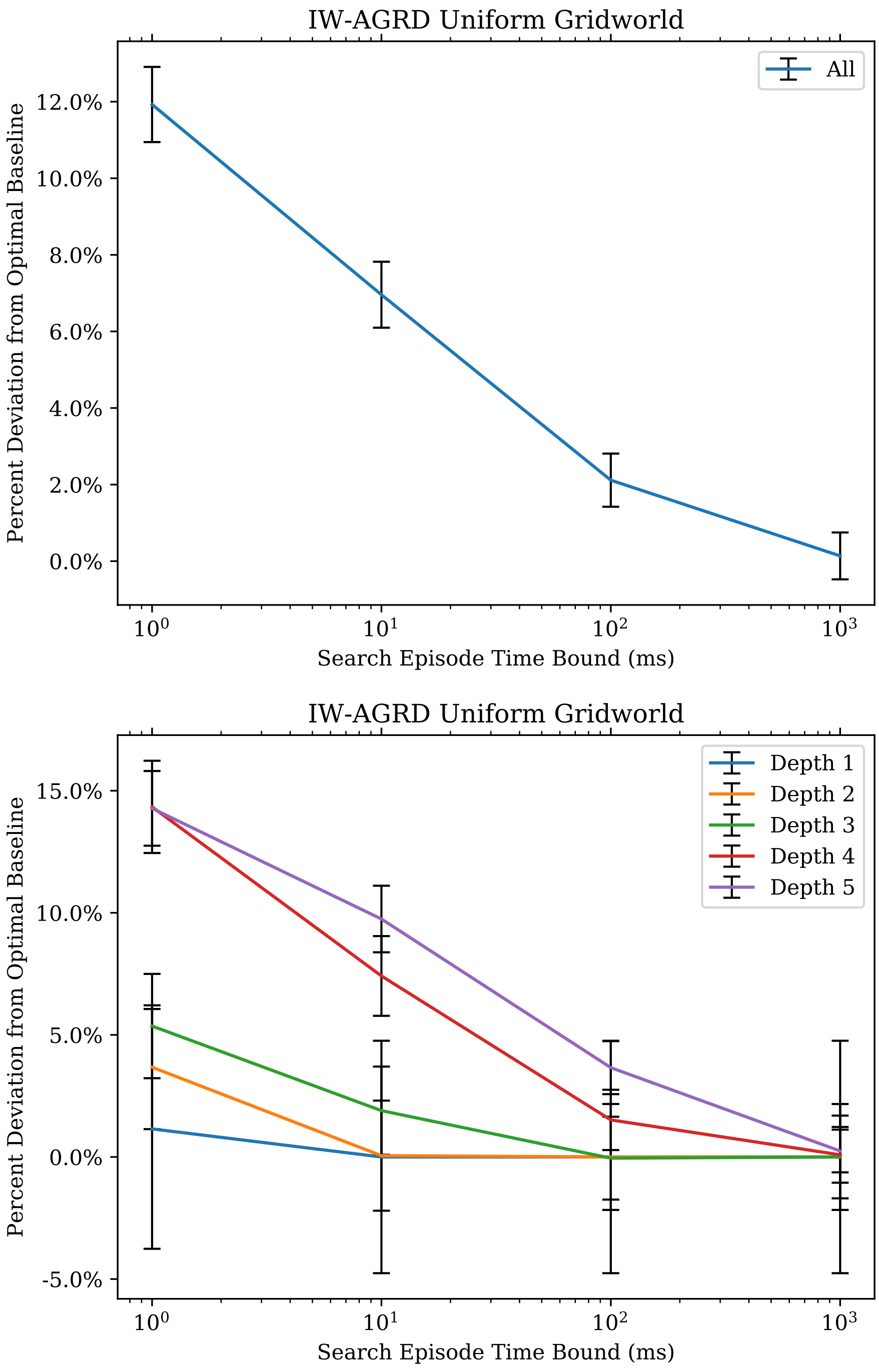

Figure 6.1: Quality of suboptimal solution relative to optimal baseline on Uniform Grid instances. Top: All experiments average results. Bottom: Results broken out by the depth bound of the instance.

no interventions that could possibly have improved detection of the subject's actual goal in that domain instance, so the observer was inactive. These experiment configurations

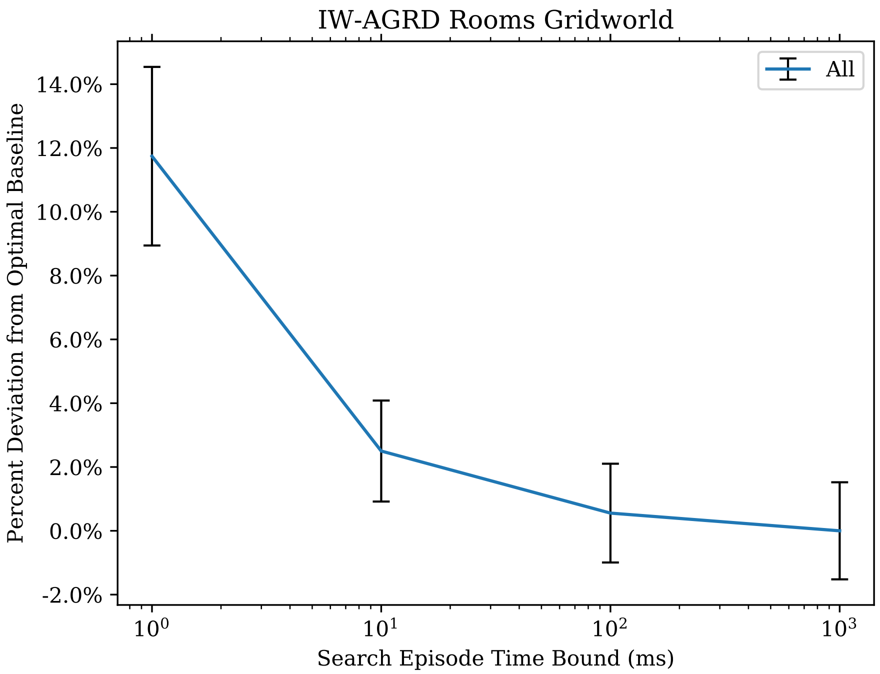

Figure 6.2: Quality of suboptimal solution relative to optimal baseline on Room Grid instances. All experiments average results.

passed through our filter because we only filtered out domain instances where there was no intervention that could affect the detection of *any* goal. We decided to retain instances where detection could be improved for at least one goal, even if no improvements are possible for other goals. In the context of our suboptimal comparison, configurations with inactive observers are not informative and only serve to make the suboptimal algorithm appear to perform better, since its deviation from the optimal baseline is 0 for those instances. Note that we only exclude configurations where detection could not be improved for every seed in both optimal and suboptimal experiments. Other configurations of the same instance with different subject goals are reported if $\psi < 1.0$ for either the baseline or achieved ψ. We also excluded from this analysis any domain instances where *IW-AGRD* could not even compute the Bayesian update for subject actions resulting from the *Identity* intervention within the time bound. Though the timeouts may have occurred in only the smallest time bounds on the most difficult instances, we do not report on those instances here to keep our comparisons across domains and time bounds fair.

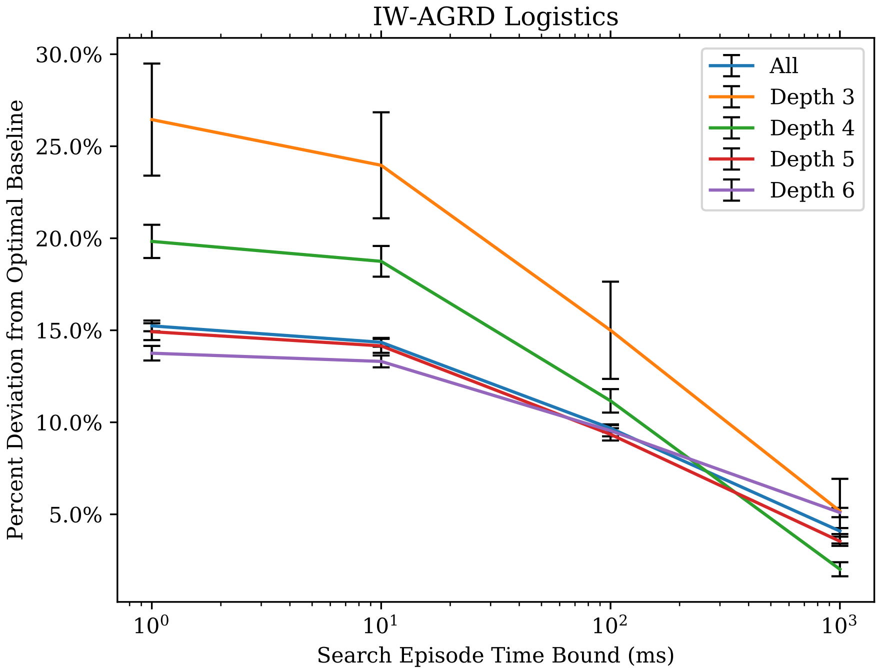

Figure 6.3: Quality of suboptimal solution relative to optimal baseline on Logistics instances. Results broken out by the depth bound of the instance.

Figures 6.1, 6.2, and 6.3 plot percent deviation from the baseline as a function of the time bound for each search episode of *IW-AGRD*. We plot both overall performance for all domains and performance broken out by depth bound in Uniform Grids and Logistics domains. Recall that the depth bound is identified as the cost of the second most expensive goal (Theorem 3, Page 24). Figure 6.1 plots results in Uniform Grid World instances. The confidence intervals for the results broken out by depth bound are very wide, so for clarity we plot the overall performance across all instances and the performance by depth bound separately in Figure 6.1-top and Figure 6.1-bottom respectively. Figure 6.2 plots only the overall performance for Rooms Grid instances. Due to the limited number of domain instances that passed filtering, the confidence intervals for results broken out by depth bound made that analysis uninformative. Figure 6.3 plots the results in Logistics instances. We plot the overall results alongside the results by depth bound to provide context for the overall quality of the algorithm. Note that we obtained many fewer instances with smaller depth bounds (an order of magnitude fewer Depth 3 vs Depth 6), which resulted in some wider confidence bounds.

The results empirically demonstrate that *IW-AGRD* exhibits the desirable property of anytime algorithms: the more time that is allocated to each search episode, the better the solution quality. The gains seen in the Logistics domain are less dramatic than in either of the Grid domains, though this is expected since the Logistics instances we experimented with are more difficult, taking up to 10 seconds to compute an optimal solution.

In the plots that break out performance by depth bound, we see a surprising difference between Gridworld (Figure 6.1-bottom) and Logistics (Figure 6.3). In Gridworld, instances with a higher depth bound lead to higher deviation from the optimal baseline, i.e. lower quality solutions, especially when the time bound is very small. In Logistics, we see the opposite result: lower depth bounds lead to lower quality solutions at low time bounds. We suspect that this discrepancy results from differences in the domain dynamics. The interconnected nature of grids means that it is easy for the observer to change course if the subject takes an action that makes blocking some cell irrelevant. The logistics location network has a different structure that may require more "route planning" through the network to reach locations the observer wishes to block access to. Committing toward one location could therefore be difficult to recover from if the subject takes actions that require the observer to pursue a different intervention. At lower depth bounds, the observer has less time to recover from moving in an ultimately fruitless direction, so the solution quality degrades when compared with higher depth bound instances.

Due to this discrepancy, we decided to examine solution quality against another metric: the fraction of the time bound divided by the total time it took the optimal solution to exhaustively explore the tree. Depth bound is correlated to runtime as discussed in Section 5.2, but a more precise comparison between the times available to make decisions will better demonstrate how practical *IW-AGRD* is for larger domains by determining if it can maintain high quality solutions when allocated only a small fraction of the time granted to the optimal algorithm.

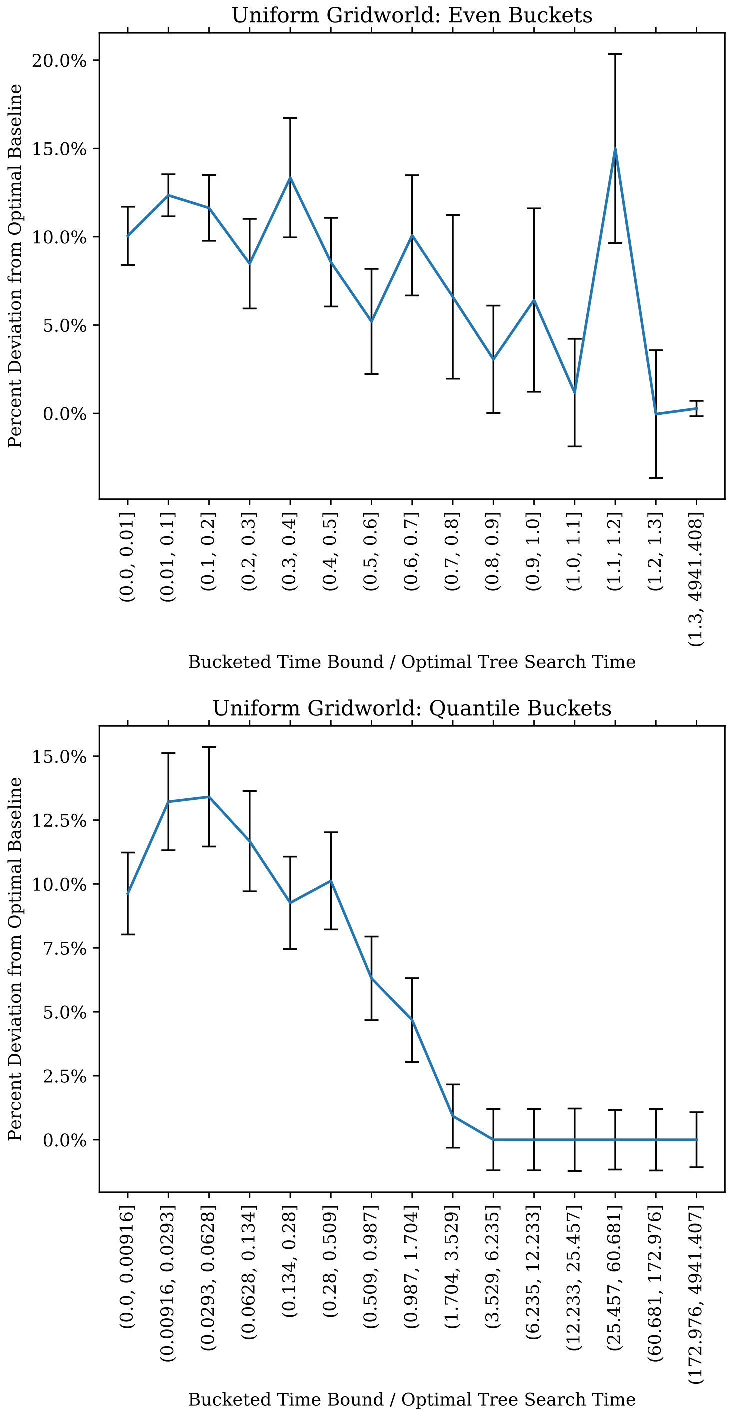

Figure 6.4: Solution quality for Uniform Grids as a function of the fraction of time in a search episode compared with *Opt-AGRD*. Top: evenly-sized buckets. Bottom: quantile buckets.

Figures 6.4 and 6.5 plot solution quality against the relative time given to *IW-AGRD* compared with *Opt-AGRD* in the Uniform Gridworld and Logistics domains respectively.

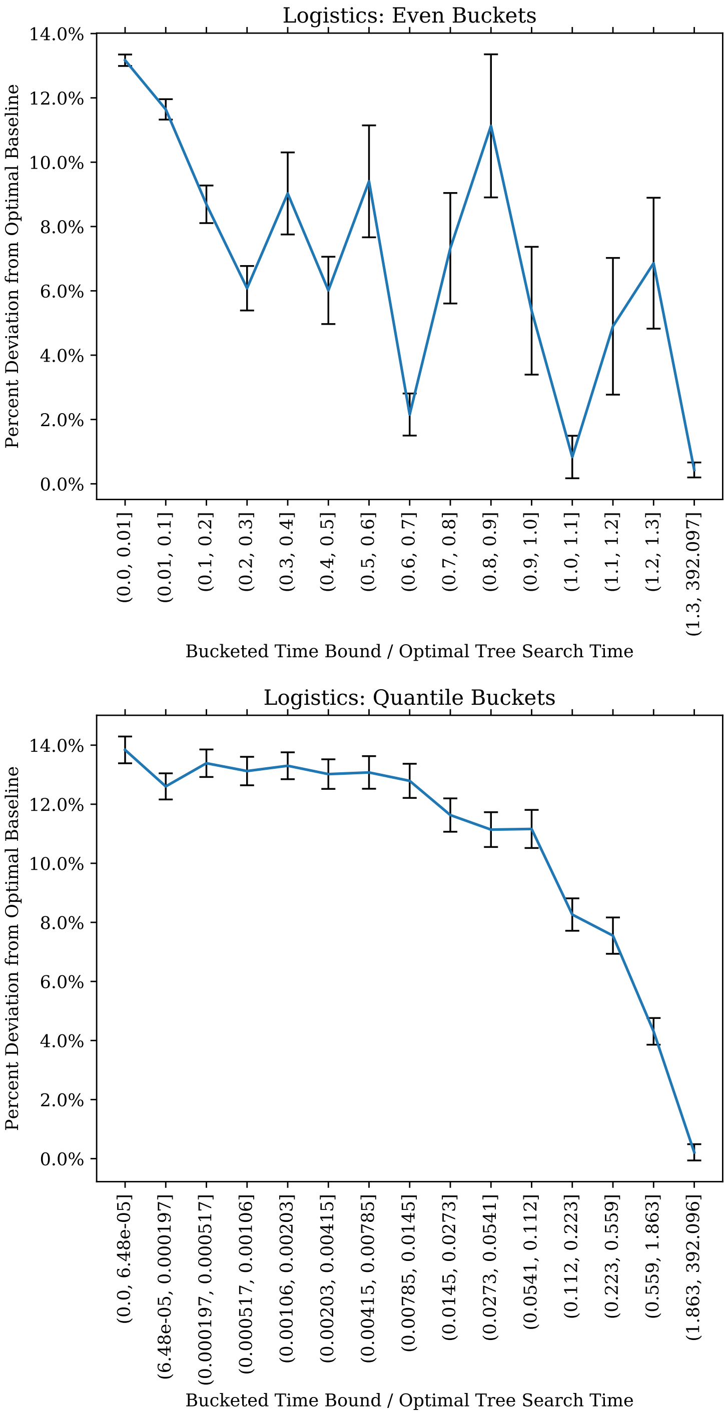

Figure 6.5: Solution quality for Logistics as a function of the fraction of time in a search episode compared with *Opt-AGRD*. Top: evenly-sized buckets. Bottom: quantile buckets.

The Rooms Gridworld experiments were uninformative because there were not enough instances to bucket the data in any meaningful way, so they have been excluded. The x-axis is computed by first taking the runtime for *Opt-AGRD* to exhaustively explore the tree as presented in Figure 5.4, grouped by domain instance. Note that we do not need to group by actual subject goal or random seed since the first search episode will be identical because the subject has taken no actions yet. For each *IW-AGRD* experiment, we divide the time bound by the optimal runtime for the corresponding domain instance, producing the fraction of time allotted to *IW-AGRD* compared with *Opt-AGRD*. For the plots, we place this fraction into buckets so that we can average the quality within a bucket. Each bucket represents a range of fractions.

We include 2 plots per domain type. In both figures, the top plot uses buckets that are evenly spaced out within a critical zone of interest: between .1 and 1.3, which corresponds to setting the time bound for *IW-AGRD* at 10%-130% of the time it takes to exhaustively explore the tree optimally. We also add buckets to capture values at the edges of our zone of interest. This produced a somewhat erratic graph with wide confidence intervals, so the bottom plot of both figures is the same data split into 15 even-width quantile buckets where each bucket has approximately the same number of data points. We'll first note that in the quantile buckets, the majority of instances are well outside our zone of interest. This is more prevalent in Figure 6.5 because the Logistics experiments were given the same set of time bounds even though the Logistics instances were orders of magnitude more difficult than Gridworld. Nevertheless, the quantile buckets provide us a clearer view of the trends in our data.

In both domains, the smallest ranges correspond to domains where *IW-AGRD* was allocated so little time relative to the size of the problem that it could do nothing but return the identity intervention. The quantile plots only start to improve solution quality as *IW-AGRD* is allocated a little less than 20% of the time that the optimal solution took. It is curious that the first bucket of Figure 6.4-bottom displays better solution quality than the next 3 despite representing the smallest fraction of time. We suspect this is an artifact of some property of the grid domains we experimented with, but we were unable to identify a root cause for this behavior. Despite the wide variance, the even buckets in the Uniform Gridworld experiements plotted in Figure 6.4-top suggest a linear relationship between the fraction of time granted to *IW-AGRD* and solution quality. It would be difficult to make the same claim with the Logistics even bucket plot because of the even wider variance present in Figure 6.5-top. What is clear is that it requires greater than 130% of the time to compute the optimal solution for the Logistics experiments to reliably recover more than 50% of solution quality that was lost at lower fractions of time.

It is unsurprising that the suboptimal algorithm would require more time to converge since it must iteratively explore smaller subsets of the full tree several times before arriving at the optimal solution. However the results suggest that the degredation in solution quality produced by *IW-AGRD* has a linear relationship with the time allotted to it as a fraction of the total time it takes to solve the problem. This makes sense, since *IW-AGRD* only prunes actions, which affects the branching factor of the tree, but not necessarily the depth. The results also support Lemma 3, which states that the depth of the tree T_{IW} has the same upper bound as T_{e_ρ}. This implies that Corollary 1 holds for T_{IW}, and so exploring it should be asymptotically very similar to T_{e_ρ}, i.e. exponential in the depth.

6.4 Discussion

IW-AGRD is a straightforward relaxation of the optimality guarantee made by *Opt-AGRD*. It provides a useful basis for analysis of a suboptimal algorithm that maintains the same restrictions as *Opt-AGRD*, namely that the observer cannot change the cost to any goal. The fact that it is an anytime algorithm makes it more widely applicable in online interactive settings than its optimal counterpart, and opens the door to applications in larger-scale domains.

However, our experiments have also exposed the weakness of this simple method. Solution quality is very low until the time bound *IW-AGRD* is given reaches a significant percentage, approximately 20%, of the time it takes to compute the optimal intervention. Ultimately, our technique is still beholden to the exponential nature of exhaustive tree exploration that it inherits from *Opt-AGRD*. As such, it can also benefit from the enhancements described in Section 5.3, but we believe there is a limit to the amount this kind of enhancement can accomplish.

6.4.1 Iterative Deepening AGRD

We have established with 2 algorithms that the depth of the search tree is the dominating factor in determining the runtime of an exhaustive tree search. We therefore lay out our vision for a suboptimal variant of *Opt-AGRD* that bounds the depth of the tree explored, though the implementation and detailed analysis is left for future work.

For the simplicity of this overview, let us continue to maintain the same assumptions about neutral observers and subject behavior that have guided the design of *Opt-AGRD* and *IW-AGRD*. Let us consider a new algorithm, *Iterative Deepening AGRD* (*ID-AGRD*), that is allowed to produce suboptimal interventions as long as it completes within a given time bound. We will borrow the basic structure of iterative exploration described for *IW-AGRD*, but instead of a widening parameter w, we use a deepening parameter κ. (We continue to use κ in place of d when referring to depth to avoid confusion with notation used for distinctiveness.)

The key problem to solve in coming up with an iterative deepening algorithm is how to evaluate the score of a non-terminal leaf node, i.e. a node that is at the depth limit, but whose goal hypothesis still has multiple goals. Let us define a generic objective function $\zeta : S \to \mathbb{R}$ that maps a state to a real number. ζ behaves exactly as ρ, except that it can operate on non-terminal leaf nodes. We will only describe ζ^{ψ}, an analogous function to ρ^{ψ} (Definition 11, Page 21), to illustrate the algorithm, but we note that other ρ functions can also be adapted into analogous ζ functions. Discussion of the requirements for such adaptation is left for future work exploring this algorithm in depth.

Our idea is to use the entropy of the goal hypothesis to help us approximate how long it will take to disambiguate each goal if we were to continue exploring the tree.

Definition 13 $\zeta^{\psi} : S \rightarrow [0,1]$ *returns an approximation of the expected fraction of the subject's plan that will have been executed when their goal is distinct. For a state* s*, let:*

$$entropy(s) = -\sum_{p_i \in P_{H_s}} p_i log(p_i) \qquad \textit{Entropy of goal posteriors}$$

$$r_i = |\phi^*_{s,g_i}| \qquad \textit{Remaining steps to reach goal } g_i \in P_{H_s}$$

Define:

$$\zeta^{\psi}(s) = \mathop{\mathbb{E}}_{p_i \sim P_{H_s}} \left[\frac{length_{subj}(s) + entropy(s)\, r_i}{length_{subj}(s) + |\phi^*_{s,g}|} \right] \tag{6.2}$$

ζ^{ψ} differs from ρ^{ψ} in 2 key ways. First, it is able to return a non-zero value for states whose goal hypothesis is greater than one. Second, we change the calculation of the fraction to be able to handle additional goals. We colored differences with Equation 4.7 in blue to highlight where ζ^{ψ} deviates from it, namely adding the entropy times remaining steps and taking an expectation distributed by goal posteriors. For each remaining goal in the hypothesis, take the remaining steps to achieve that goal and multiply it by the entropy of the goal posteriors in that state. This term is added to the length of the subject's plan to reach s, giving us an approximation of when we would expect to distinguish that particular goal. Dividing this by the total steps to achieve the goal optimally yields an approximation of the fraction of the subject's plan that will be executed prior to distinguishing that goal. Intuitively, if the entropy is higher, we are less certain of which goal the subject is pursuing, so we will estimate that it will take longer to distinguish their goal. The final score of the non-terminal node is the weighted average over all goals using the goal posteriors.

To determine the best intervention of a state given, we define e_{ζ}, an analogous function to e_{ρ} (Definition 10, Page 19).

Definition 14 $e_{\zeta} : S \times \mathbb{Z}^{\geq 0} \rightarrow \mathbb{R}$ *is a function that computes the expectation of the best* *achievable ζ of a given state for a given scoring function ζ and a given depth bound κ:*

$$e_\zeta(s,k) = \begin{cases} \zeta(s) & \textit{if } k = \kappa \textit{ or } |H_s| \leq 1 \\ \underset{a_{obs} \in A_{obs}(s)}{\mathit{min}} \mathit{score}\big(a_{obs}(s), k\big) & \textit{otherwise} \end{cases} \tag{6.3}$$

$$\mathit{score}(s,k) = \underset{a_{subj} \sim P_s}{\mathbb{E}} \Big[e_\rho\big(a_{subj}(s), k+1\big) \Big] \ . \tag{6.4}$$

$e_\zeta(s,k)$ and the altered *score* function are then used to expand the state space tree, approximating all exploration that would descend below depth 2κ. Once the algorithm completes its exploration of the tree with bound κ, we increment κ so that trees of increasing depth can be explored until time runs out.

Using the entropy of the posteriors in this way exhibits some helpful qualities. High-entropy posteriors, i.e. those with many goals that are equally likely, will be estimated to take longer to disambiguate. Conversely, low-entropy posteriors will be estimated to take little time to disambiguate. This rewards states where one goal is much more likely than the rest, which are indeed states that we intuitively want to drive the subject toward. However it is unclear whether this algorithm would prefer non-terminal leaf nodes with low entropy over terminal leaf nodes representing states where the goal has been fully disambiguated. We may require a mechanism (reminiscent of UCT [Kocsis and Szepesvári, 2006]) that tracks our confidence in the score of a branch based on the proportion of non-terminal leaf nodes vs terminal leaves. This question and others about the behavior of this proposed algorithm are promising research directions.

CHAPTER 7

Conclusion

As AI-driven systems become widespread, it is crucial to extend our models of multiagent interaction. In this thesis, we have presented Active Goal Recognition Design, a new problem setting that models observers with agency to aid goal recognition online. AGRD adapts the conventional Goal Recognition Design problem to online settings where the observer and subject interleave actions, extending GRD to handle entirely new classes of problems that require reasoning about the passage of time in an evolving system.

The thesis we pursued in this research is:

> Active Goal Recognition Design is a useful new problem setting that sits between passive Goal Recognition and two-player games.

- In Chapter 1, we motivated our research and introduced our approach.
- In Chapter 2, we described prior work, highlighting the growing research on active observers influencing subject agents online.
- In Chapter 3, we discussed the conventional objective functions for GRD based on distinctiveness. We provided counter-examples to illustrate their limitations, and proposed ψ, an objective that incorporates reaction time and balances resource allocation between recognition efforts for different goals. We also discussed challenges with GRD systems in online settings.
- In Chapter 4, we defined the formal problem setting for Active Goal Recognition Design and provided theoretical proofs on important characteristics of this problem.

- In Chapter 5, we proposed *Opt-AGRD*, an algorithm to optimally solve an AGRD problem instance that exhaustively explores the state space tree induced by simulating subject and observer actions as the system evolves. We proved and demonstrated empirically that the cost of this exploration is exponential.
- In Chapter 6, we proposed *IW-AGRD*, an anytime suboptimal variant of *Opt-AGRD*, that aggressively prunes subject actions from the tree to reduce the number of branches explored. We evaluated this variant and discussed enhancements and research directions to carry this work forward beyond its current limitations.

We conclude, given the limitations of previous work and the breadth of applicability of AGRD, that AGRD is a useful new problem setting.

7.1 Related Work

The AGRD problem setting captures a level of interaction between subjects and observers not previously modeled in the context of Goal Recognition or GRD. We now discuss closely related approaches and how our formulation improves on prior work.

7.1.1 Active Observers and GRD

In Active Goal Recognition (AGR) [Shvo and McIlraith, 2020], the observer is granted agency to take sensing actions that facilitate observations of the subject to aid goal recognition. Though their evaluation uses ψ as we've defined it to report on the quality of their solution, they do not directly minimize this objective. They instead plan their actions to maximally prune the goal hypothesis with the heuristic guidance that pruning more goals at a time is better. However, this planning procedure does not consider how the system may evolve in two key ways. Firstly, if there is a choice between an intervention that distinguishes goal A and another that distinguishes goals B and C, AGR will always select the one that distinguishes 2 goals, even if the first intervention is the observer's only opportunity to distinguish goal

A before it is achieved. Secondly, as we discussed in Section 3.2, the assumption that the observer will have time to execute their plan without modeling the evolving system is problematic in domains with time pressure.

The formulation of Goal Elicitation Planning [Amos-Binks and Cardona-Rivera, 2020] is more closely related to our own in that their model interleaves subject and observer actions. However they do not explicitly model the *relationship* between subject and observer actions, but rather choose from a precomputed set of static observer plans based on whether the plan can possibly be achieved in time to affect the subject. Their myopic focus on this plan selection precludes them from planning for sequences of interventions that incrementally reduce wcd. Additionally, by focusing on wcd, their observer plans can only reduce the worst case. Once the subject transitions to a branch of the state space that is not worst case, their observer will be inactive, even if an intervention exists that could further reduce the length of the most non-distinct path remaining to the subject.

In contrast to these approaches, we have presented a problem formulation closer to online planning. By continuously minimizing the expectation of the objective, we allow our observer to react to an evolving world. The AGRD formulation is applicable to a broader class of problems than prior work, and we hope the algorithms and analysis we have presented can provide a baseline for further research.

7.1.2 Inverse Reinforcement Learning

Goal Recognition bears similarity to Inverse Reinforcement Learning. IRL is a problem setting where a system attempts to recover the reward function of a subject (often referred to as an "expert") acting within an MDP. The reward function can be analogous to the goal of an agent, especially in the MDP we introduced in Section 4.5 where the observer only receives reward when the subject's goal is known. In particular, Ramachandran and Amir [2007] introduce a method of learning reward functions as a sequence of Bayesian updates from observations and priors over a finite set of possible reward functions. But while there are similarities, IRL focuses on generalizing knowledge gained through learning a reward function. Our work in goal recognition focuses on isolated inference of a subject's goal that is not intended to generalize to the next problem instance. An interesting direction could be to explore using IRL to help learn the "preferences" of subjects within specific domains. These preferences could translate to weighting certain actions as more likely than others, thus allowing the observer to strategize better with additional information. We leave this exploration for future work.

7.2 Limitations

Our approach is not without drawbacks. As our evaluation demonstrates, our simple optimal algorithm, *Opt-AGRD*, is impractical for large domains since simulating every possible branch of the search tree is exponential in the depth of the tree. The suboptimal algorithm we presented, *IW-AGRD*, provides anytime behavior to mitigate this scaling issue at the cost of solution quality. However, its exploration of the reduced search tree inherits the same fundamental flaw of exhaustively evaluating the tree, meaning solution quality degrades to inactivity unless each search episode of $IW\text{-}AGRD$ is allowed to run for a significant fraction of the time it takes to compute an optimal solution. We discussed some possible variations and enhancements to improve performance in Section 5.3.

7.3 Possible Extensions

As discussed in the evaluations in Sections 5.2 and 6.3, *Opt-AGRD* and $IW\text{-}AGRD$ have scaling issues due to the exponential nature of the trees they explore. In Section 6.4.1, we proposed another variant, *ID-AGRD*, that would directly address the scaling issue by exploring trees of reduced depth. Implementing and evaluating this method is a compelling direction for future work. We also note that adapting approximate MDP methods, such as MCTS, may be beneficial in computing quality solutions in a scalable way.

This work makes several restrictive assumptions, chief among them that the subject acts optimally and that the observer cannot change the optimal cost to any goal that has not been pruned yet. Our formulation can be extended to suboptimal subjects, for instance by adapting the work on bounded suboptimality in GRD [Keren *et al.*, 2015]. In addition, we could allow the observer to change optimal plan costs from their original values within a constant bound. These modifications would unfortunately only increase the size of the tree explored by *Opt-AGRD*, so would have to be examined in tandem with other work (e.g. as described above) to address the scalability of AGRD algorithms.

Lastly, a central (and costly) assumption made by this work is that all possible subject plans are equally likely. This assumption requires us to compute all plans to active goals at every node in T_{e_ρ} that represents a successor of an intervention. It would be interesting to explore algorithms that can repair our representation of all plans rather than recomputing the whole set. But more importantly, it would be useful to explore other probability distributions for subject actions. For instance, we could use a Boltzman-like distribution to model "noisy rationality" as in [Fisac *et al.*, 2019; Ramachandran and Amir, 2007]. However, without re-computing plans to goals after executing an intervention, we do not know if or by how much the cost to any goal has changed. Employing different distributions would require us to more efficiently compute this information or otherwise relax that restriction.

LIST OF REFERENCES

[Amos-Binks and Cardona-Rivera, 2020] Adam Amos-Binks and Rogerlio E. Cardona-Rivera. Goal elicitation planning: Acting to reveal the goals of others. *Proceedings of Advancements in Cognitive Systems*, 2020.

[Ang *et al.*, 2017] Samuel Ang, Hau Chan, Albert Xin Jiang, and William Yeoh. Game-theoretic goal recognition models with applications to security domains. In *International Conference on Decision and Game Theory for Security*, pages 256–272. Springer, 2017.

[Avrahami-Zilberbrand and Kaminka, 2014] Dorit Avrahami-Zilberbrand and Gal A Kaminka. Keyhole adversarial plan recognition for recognition of suspicious and anomalous behavior. *Plan, Activity, and Intent Recognition: Theory and Practice*, pages 87 – 121, 2014.

[Bellman, 1957] Richard Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[Bisson *et al.*, 2011] Francis Bisson, Froduald Kabanza, Abder Rezak Benaskeur, and Hengameh Irandoust. Provoking opponents to facilitate the recognition of their intentions. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[Boddy *et al.*, 2005] Mark S Boddy, Johnathan Gohde, Thomas Haigh, and Steven A Harp. Course of action generation for cyber security using classical planning. In *ICAPS*, pages 12–21, 2005.

[Bonet and Geffner, 2003] Blai Bonet and Hector Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proc. ICAPS 2003*, 2003.

[Bui, 2003] Hung Hai Bui. A general model for online probabilistic plan recognition. In *IJCAI*, volume 3, pages 1309–1315, 2003.

[Cormen *et al.*, 2009] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, 2009.

[Edelkamp *et al.*, 2009] Stefan Edelkamp, Carsten Elfers, Mirko Horstmann, Marcus-Sebastian Schröder, Karsten Sohr, and Thomas Wagner. Early warning and intrusion detection based on combined ai methods. In *19th International Conference on Automated Planning and Scheduling*, 2009.

[Fisac *et al.*, 2019] Jaime F Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S Shankar Sastry, and Anca D Dragan. Hierarchical game-theoretic planning for autonomous vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9590–9596. IEEE, 2019.

[Ginsberg and Harvey, 1992] Matthew L. Ginsberg and William D. Harvey. Iterative broadening. *Artificial Intelligence*, 55:367–383, 1992.

[Ha *et al.*, 2011] Eun Young Ha, Jonathan P Rowe, Bradford W Mott, and James C Lester. Goal recognition with markov logic networks for player-adaptive games. In *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.

[Kabanza *et al.*, 2010] Froduald Kabanza, Philipe Bellefeuille, Francis Bisson, Abder Rezak Benaskeur, and Hengameh Irandoust. Opponent behaviour recognition for real-time strategy games. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[Keren *et al.*, 2014] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.

[Keren *et al.*, 2015] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design for non-optimal agents. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[Keren *et al.*, 2016a] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design with non-observable actions. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[Keren *et al.*, 2016b] Sarah Keren, Avigdor Gal, and Erez Karpas. Privacy preserving plans in partially observable environments. In *IJCAI*, pages 3170–3176, 2016.

[Keren *et al.*, 2020] Sarah Keren, Haifeng Xu, Kofi Kwapong, David C Parkes, and Barbara Grosz. Information shaping for enhanced goal recognition of partially-informed agents. In *AAAI*, pages 9908–9915, 2020.

[Knuth and Moore, 1975] Donald E Knuth and Ronald W Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975.

[Kocsis and Szepesvári, 2006] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning (ECML-06)*, pages 282–293, 2006.

[Masters and Sardina, 2019] Peta Masters and Sebastian Sardina. Cost-based goal recognition in navigational domains. *Journal of Artificial Intelligence Research*, 64:197–242, 2019.

[Mirsky *et al.*, 2018] Reuth Mirsky, Roni Stern, Kobi Gal, and Meir Kalech. Sequential plan recognition: An iterative approach to disambiguating between hypotheses. *Artificial Intelligence*, 260:51–73, 2018.

[Oh *et al.*, 2014] Jean Oh, Felipe Meneguzzi, and Katia Sycara. Probabilistic plan recognition for proactive assistant agents. *Plan, Activity, and Intent Recognition: Theory and Practice*, pages 275 – 288, 2014.

[Pozanco *et al.*, 2018] Alberto Pozanco, E Yolanda, Susana Fernández, and Daniel Borrajo. Counterplanning using goal recognition and landmarks. In *IJCAI*, pages 4808–4814, 2018.

[Ramachandran and Amir, 2007] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

[Ramírez and Geffner, 2009] Miquel Ramírez and Hector Geffner. Plan recognition as planning. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, pages 1778–1783, 2009.

[Ramírez and Geffner, 2010] Miquel Ramírez and Hector 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)*, pages 1121–1126, 2010.

[Shoham and Leyton-Brown, 2009] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations.* Cambridge University Press, 2009.

[Shvo and McIlraith, 2020] Maayan Shvo and Sheila A McIlraith. Active goal recognition. In *AAAI*, pages 9957–9966, 2020.

[Sturtevant, 2012] Nathan Sturtevant. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games*, 4(2):144–148, 2012.

[Sukthankar *et al.*, 2014] Gita Sukthankar, Christopher Geib, Hung Hai Bui, David Pynadath, and Robert P Goldman. *Plan, Activity, and Intent Recognition: Theory and Practice.* Elsevier, 2014.

[Wayllace *et al.*, 2017] Christabel Wayllace, Ping Hou, and William Yeoh. New metrics and algorithms for stochastic goal recognition design problems. In *IJCAI*, pages 4455–4462, 2017.