THESIS

REDUCING GOAL STATE DIVERGENCE WITH ENVIRONMENT DESIGN

Submitted by

Kelsey Sikes

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2025

Master's Committee:

Advisor: Sarath Sreedharan

Nathaniel Blanchard
Edwin K.P. Chong

ABSTRACT

REDUCING GOAL STATE DIVERGENCE WITH ENVIRONMENT DESIGN

At the core of most successful human-robot collaborations is alignment between a robot's behavior and a human's expectations. Achieving this alignment is often difficult, however, because without careful specification, a robot may misinterpret a human's goals, causing it to perform actions with unexpected, if not dangerous side effects. To avoid this, I propose a new metric called Goal State Divergence ($\mathcal{GSD}$), which represents the difference between the final goal state achieved by a robot and the one a human user expected. In cases where $\mathcal{GSD}$ cannot be directly calculated, I show how it can be approximated using maximal and minimal bounds. I then leverage $\mathcal{GSD}$ in my novel human-robot goal alignment design (HRGAD) problem, which identifies a minimal set of environment modifications that can reduce such mismatches. To illustrate the effectiveness of my method for reducing goal state divergence, I then empirically evaluate it on several standard planning benchmarks.

ACKNOWLEDGEMENTS

I would like to begin by thanking my amazing advisor, Dr. Sarath Sreedharan, for the opportunity to work on these problems and for investing so much into me as a researcher. Working on this project has been a very challenging and rewarding experience, and under your supervision, my knowledge and skills have grown tremendously. Thank you for your patience and guidance in helping me bring this work to fruition, I've enjoyed every discussion and technical debate along the way.

To Dr. Nathaniel Blanchard, thank you for supervising this work and for all your help and advice during my graduate school journey. Thank you, Dr. Edwin K. P. Chong, for agreeing to be part of my thesis committee and for bringing your wealth of knowledge to the supervision of this work. Special thanks to Dr. Sarah Keren for your feedback, invaluable insights, and passion for this area of research – I am deeply grateful for having had the opportunity to learn from and co-author a paper with you.

To everyone in the HAPI Lab, including Turguy Caglar, Erfan Mirhaji, Septia Rani, Dennis Kim, Phil Hopkins, Brittany Cates, Trisha Ghali, and Malek Mechergui, I am incredibly grateful for your support. Thank you all for being such fun, intelligent people who I get the privilege of being around and learning from daily.

To all the amazing people I've met since moving to Colorado, including Mina Roueinfar, Mik Hammers, and Shannon Stancu-Goldberg, thank you all for being such wonderful people who've helped make Fort Collins feel like home. To my family and friends, thank you all for your unwavering support and belief in me – you've been the best support network a graduate student could ask for. And lastly to Maggie, the coolest Australian Shephard ever, thank you for being such an amazing dog. I will never not be obsessed with you.

DEDICATION

To my family, friends, and all those who

have fun doing hard things well.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
INTRODUCTION

In recent years, due to continued and rapid advancements in the field, Artificial Intelligence (AI) has become one of the most adopted technologies ever. With this ubiquity has come a greater proliferation of human-robot interactions, as society's reliance on AI continues to grow. For these collaborations to truly thrive, however, robots must improve their ability to generate behavior that satisfies the goals or desired outcomes of humans in a way that they would anticipate. Despite the need for this, consistently achieving this alignment in real world systems is difficult.

When a human interacts with a robot, they begin with a model of what they believe the robot's current state and capabilities to be (i.e. the human's belief model of the robot). Based on these beliefs, the human will assign the robot a *goal* – a specific, measurable target to be accomplished. To achieve this objective, the robot will need to reach a *goal state* – an ending configuration where all underlying criteria for its given objective has been achieved. In varying levels of detail, this criteria will be communicated to the robot by the human and implied as part of its objective.

Accompanying this goal state, will be the human's beliefs about what behaviors they think the robot will exhibit to achieve its goal, and what the state of the world will be after its completion. This set of assumptions is the *human's expectations*. Upon reaching a goal state, ideally the robot would have accomplished its goal in a way expected by the human. But due to expectation mismatches, this often isn't the case.

Rather, because the human's expectations are based off their belief model of the robot, they may not match true reality (i.e. the robot's actual model). This means the robot may possess capabilities the human was not aware of, lack capabilities they assumed it had, or be operating in a state completely different from the human's own state. As a result,

when specifying a given objective, instructions that seemed clear to the human may be misinterpreted by the robot or inexecutable in its model.

Further contributing to this knowledge asymmetry could be a variety of other factors, including limited inferential capabilities, cognitive biases, or a lack of experience interacting with robots on the part of the human. As a result, the robot may react to the human's instructions by performing actions which lead it to a goal state radically different from what the human expected, apart from anything clearly included in their goal specification. This difference in expectations may produce undesirable side effects, which in benign cases might frustrate the human but in more serious ones could threaten their safety. For this reason, verifying that the final goal state a robot is trying to satisfy aligns with a human's expectations, is of paramount importance.

To solve this problem, previous work has largely focused on aligning a robot's plan with a human's expectations to resolve expectation mismatches, using techniques like value alignment [1], explicable behavior generation [2] and model reconciliation [3]. However, such approaches largely ignore the final goal states a robot may achieve. This means though a human may understand why a robot is performing certain actions towards a goal, these methods don't guarantee that it will achieve its goal in a human-expected way or avoid unwanted side effects in the process. In this document, I address this problem by examining issues that may arise when the potential goal states a human expects a robot to achieve, differ from those it might achieve.

To do this, I introduce Goal State Divergence ($\mathcal{GSD}$), a novel metric I developed to measure the difference between the final goal state a robot achieves, and the one a human expected the robot to achieve. For a given goal specification, I then explore how environment design [4] can be used to avoid the potential expectation mismatches that lead to this divergence. Specifically, I identify viable design modifications that can be made to a robot's model, leading it to a final goal state closely aligned with a human's expectations. These

identified designs are then implemented by adding or removing elements from the robot's model, reducing their $\mathcal{GSD}$. This reduction allows the robot to accomplish its goal in a more human-expected way, better fulfilling the desires of the human while avoiding unanticipated side effects in the process.

Because multiple plans may satisfy a goal state or the plan a human expects a robot to execute may be unknown, directly calculating $\mathcal{GSD}$ in many situations is impossible. To account for this, I show how $\mathcal{GSD}$ might be approximated using maximal and minimal bounds, which I identify using novel classical planning-based compilations for each given environment design problem. Once calculated, I determine potential design modifications that if made, will minimize this estimated value.

In summary, my specific research contributions include:

- The introduction of Goal State Divergence ($\mathcal{GSD}$), a novel metric for measuring the difference between a human and robot's final goal states.
- The development of a maximal and minimal bound for approximating $\mathcal{GSD}$ in cases where it can't be directly calculated, and a demonstration of how to calculate each.
- The creation of a novel design problem which uses these approximations to reduce $\mathcal{GSD}$.
- A comprehensive empirical evaluation of our $\mathcal{GSD}$ method on a set of standard IPC planning benchmarks.

Chapter 2
RELATED WORKS

In this section, I briefly discuss past works that inspired this research from the areas of i) environment design, ii) model reconciliation, iii) AI safety, and iv) explicable planning, and discuss the ways in which our work builds upon each.

2.1 Environment Design

Environment design involves intentionally shaping an agent's environment to produce or prevent certain behaviors by adding or removing specific elements from it (i.e. by making environment modifications). As a result, the agent can then better maximize or minimize some objective [4]–[6]. Originally, [4] introduced this method in the context of sequential decision-making tasks modeled by Markov Decision Processes, where limited incentives were used to influence an agent to perform certain policies. Since then, many different use cases of environment design, have come into existence. For example, several works have explored how design could be used to configure environments for reinforcement learning [7] or agent's that are biased decision-makers [8].

Most prominently, however, has been the use of environment design to facilitate better goal and plan recognition in planning agents [9]–[11]. Of these, most have relied on heuristic search to find an optimal set of designs including Keren *et al.* [12] who used it to maximize agent objectives in uncertain, stochastic environments and Keren *et al.* [13] who employed the same approach to find the maximum shared agent-designer utility in Equi-Reward Utility Maximizing Design (ER-UMD) settings. To avoid calculating all possible design modifications, this work was extended by Keren *et al.* [14] who limited what changes could be made to a model, then mapped each one to a dominating modification. This is significant because some design modifications will have more impact on an agent's utility then others. Thus,

by only calculating the subset of changes relevant to an agent's goals, the algorithms effectiveness is increased. This is similar to my own work, where I also identify a minimal set of design changes while meeting certain other criteria. However, whereas most research in this area has used environment design to reveal an agent's objectives, I use it to help an agent end in a final goal state as closely aligned with a human's expectations as possible.

2.2 Model Reconciliation

The goal of model reconciliation is to find alignment between a human and a robot's models by resolving inconsistencies between them. Once settled, the robot can better achieve its goals, in a way that conflicting models would prevent (i.e. the human's goals are satisfied free of confusion, unwanted side effects or human intervention). To effectively do this, many works have relied on communicative actions [15], [16] and explanation generation strategies [17] to pinpoint and reconcile model differences. For example, [3] do this by having a robot first acknowledge model discrepancies between itself and a human collaborator, then reduce them by suggesting changes to their model. Once updated, the robot then generates an optimal plan congruent with the human's modified beliefs. To describe such a plan, others have studied how certain cognitive biases and societal expectations [18] or personalization techniques [19] could be used to improve a robot's explanations.

For a robot's explanations to be effective, however, the human must be able to understand them. Given that many humans interacting with a robot will have limited inferential capabilities or lack experience dealing with AI agents, ensuring this is difficult. [20] addresses this divide by representing the human's model as a domain abstraction and having a robot give different explanations about its behavior to the human based on their level of expertise. The human's model is then updated as information that was absent from it is provided. To give inferential assistance, [20] also suggests augmenting existing model reconciliation strategies

with other approaches like casual explanations [21] or the abstractions and refutation of specific foils used in their own work.

Though such methods have proved effective, they assume the inconsistencies between the human and robot's models are known. In cases where the human's model is uncertain, incomplete, or multiple human's in the loop exist, some work has looked at using conformant explanations to achieve model reconciliation [22]. Here, the robot expresses one minimally complete explanation about its plan to annotated versions of all possible human models, allowing a common ground between them to be established. While in works like these reconciling the human and robot's models is important, the main concern of this thesis is reconciling the human and robot's final goal states. Though, in future extensions of this work, combining environment design with standard reconciliation strategies like the ones described, may prove helpful for achieving this goal.

2.3 AI Safety

For a robot to be a permanent part of any environment, its behavior must be safe (i.e. it only performs actions that won't cause harm). AI safety is a research area concerned with ensuring this happens. Though many unique problems in this space exist, designing systems to avoid negative side effects has become especially popular [23]–[26]. Such work often assumes that to reliably produce safe behavior, a robot must avoid performing any actions that correspond to a side-effect producing state. To achieve this, many works have tried aligning a robot's objectives with human values [27], using environmental knowledge to infer human preferences [28], learning from human feedback [29], [30], and penalizing a robot for any behaviors which produce a negative side effect [31]. Such work mirrors my own, which seeks to avoid unintended side effects by making the final goal state a robot achieves as identical to what a human expects as possible. However, whereas the former often

implements methods to solve specific AI safety issues, I treat them as something mitigated as a by-product of my approach.

Because side effects caused by a robot's actions may change its environment – surprising and potentially harming a human user – many works assume access to a set of locked features. Zhang, Durfee, and Singh [32] do this by allowing the robot to only modify things in its environment that it has been given explicit permission to. For all unknown features, the robot is then tasked with selectively querying the human to ask about them. To address the value misalignment problem and avoid undesirable side effects, [1] take a similar approach by having the robot query the human to model any asymmetries that exist between them. Using this information, the robot is then able to better understand the human's goals and independently achieve them. While comparable to my work, here the human and robot's models are reconciled before any attempts at achieving their goals have been made or any unwanted side effects stemming from the robot's behavior have been produced. In contrast, in my work, model reconciliation occurs after the robot has tried to achieve the human's goals but ended in a final goal state different from what they expected, potentially accompanied by unwanted side effects.

Similar to this is [33], who also have the robot query the human before carrying out its plan but specifically to avoid negative side effects (NSEs). However, in their work the human is asked to assess the safety of the robot's plan, then engage in environment shaping to help mitigate any risks associated with it. While this may avoid negative side effects, the extensive human intervention required by this method is not practical for most real-world settings. In my method, I circumvent this problem by instead having the robot modify its own model according to a specification of the human's model. This avoids the need for any human intervention while also providing the robot with instant feedback about whether or not its final goal state matches what the human wanted.

2.4 Explicable Planning

In explicable planning, the human has a model that the robot is incapable of altering. The robot therefore tries to generate plans aligned with the human's expectations, achievable in its model. In [2] this is accomplished by having a model use conditional random fields (CRFs) to learn a labeling scheme of robot actions to calculate a plan's explicability and predictability. Similarly, in [34], reconciliation search is applied to the learned distance between a robot's actual plan and a human's expected plan to guide robots towards more understandable plans. Though the methods used by [2] and [34] both result in more explicable robot plans, the former relies on a model-free approach to achieve this, while the latter does not. To narrow model disparities and produce more interpretable robot plans, many works have also tried combining explicable planning with existing reconciliation strategies like communicative actions and explanation generation [15], [35]–[37].

To avoid potential safety issues stemming from human-AI model mismatches, some work has also explored how explicable planning could be used. In [38], this is done by implementing a designer-specified safety bound which limits a robot's actions to only those guaranteed to be safe. Though our work isn't directly concerned with safety issues, it presupposes that if a robot reaches a final goal state aligned with a human's expectations, unwanted side effects will be avoided. This means in a sense we are also limiting the robot's actions (albeit more indirectly) by influencing it to execute specific plans that will result in a final goal state closely aligned with what a human expected.

To produce more interpretable robot behavior, environment design has also been applied to explicable planning. Like in [39], where a framework illustrating how this combination could be used to bring about more explicable, legible, and predictable robot behavior is presented. And in [40], where this combination is applied to structured environments to influence a human's expectations of a robot performing repetitive tasks, while promoting

more interpretable behavior. Here, the one-time cost of a design modification is weighed against the repeated cost of a robot generating explicable behavior over a time horizon, which mirrors our approach. However, whereas works like this and explicable planning methods in general focus on the specific plan and actions executed by a robot to accomplish its goals, my work does not. Rather, I solely concentrate on whether a robot's final goal state matches a human's expectations and results in its goals being achieved, irrespective of the actions or plan used to get there.

Chapter 3
BACKGROUND INFORMATION

3.1 Classical Planning

Planning consists of a robot carrying out a set of actions to achieve a goal. The robot selects these actions using a predefined model, which captures both its capabilities and the dynamics of its environment. Though such models can be expressed in numerous ways, in this work, I focus on classical planning models (often referred to as automated planning models), which identify a set of actions that allow a robot to reach a goal state given some initial state [41].

To define these models, I use the tuple $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{D}$ represents the model's domain, $\mathcal{I}$ some initial state, and $\mathcal{G}$ the desired goal state. The domain $\mathcal{D}$ corresponds to the environment the robot is operating in, which we further characterize using the tuple $\mathcal{D} = \langle \mathcal{F}, \mathcal{A} \rangle$. Here, $\mathcal{F}$ is a set of fluents or propositional, boolean state variables used to uniquely describe the state space for a given planning problem. This means that for any state s in that space, a set of unique fluents exists to represent it (i.e. $s \subseteq \mathcal{F}$ for all states s). $\mathcal{A}$ corresponds to a set of actions that, when executed, change the state of the world (i.e. the state of fluents). We define these using the tuple $a = \langle pre_{+}(a), pre_{-}(a), add(a), del(a) \rangle$, where $pre_{+}(a)$ and $pre_{-}(a)$ are positive or negative preconditions that must be satisfied for an action to be executed, and $add(a)$ and $del(a)$ are sets of add and delete effects that describe how the world has changed when an action is executed. To capture the effects of executing an action in some state, the transition function $\mathcal{T}_{\mathcal{M}} : 2^{\mathcal{F}} \times \mathcal{A} \rightarrow 2^{\mathcal{F}}$ is used, where:

$$\mathcal{T}_{\mathcal{M}}(s,a) = \begin{cases} s \cup add(a) \ \setminus \ del(a) & \text{if } exec(s,a) \\ undefined & \text{otherwise} \end{cases}$$

In this work, I also make this transition function applicable to action sequences by overloading the notation such that:

$$\mathcal{T}_{\mathcal{M}}(s, \langle a_1, ..., a_k \rangle) = \mathcal{T}_{\mathcal{M}}(...(\mathcal{T}_{\mathcal{M}}(s, a_1), ..., a_k))$$

The solution to a classical planning problem is a plan π, which is a sequence of actions $\langle a_1, ..., a_k \rangle$, whose successful execution results in an ending state that satisfies a goal condition (i.e. π is a plan if $\mathcal{T}_{\mathcal{M}}(\mathcal{I}, \pi) \supseteq \mathcal{G}$). Each action in a plan has a cost c. The sum of all these costs is then the cost of the plan, denoted by $c(\pi) = \sum_{a_i \in \pi} c(a_i)$. For a given model $\mathcal{M}$, an optimal plan π^*, is one whose cost is lower than all other plans. Because many optimal plans may exist, I denote the set of all optimal plans for a model by $\Pi^*_{\mathcal{M}}$ and the set of all plans by $\Pi_{\mathcal{M}}$.

3.2 Planning Domain and Definition Language

A planner is a program that generates a sequence of actions to move a robot from an initial state to a goal state. To do this, the planner takes in a model, then uses a representational language like STRIPS or ADL to read the information stored inside it and move the robot through its environment. In this work, I use a language often used for classical planning problems, the Planning Domain and Definition Language or PDDL, to accomplish this [42]. PDDL was first introduced in the 90s for use in the first International Planning Competition (IPC), for which it has now become the standard language [43].

To capture a planning task, PDDL relies on the use of a domain and problem file. The domain file consists of predicates, fluents and actions, and describes the domain or setting in which a robot is operating. For a given action to become executable, all its preconditions must first be satisfied. Upon execution, any add or delete effects associated with that action may then result in changes to the robot's domain. The problem file consists of objects,

some initial state, and the goal specification a robot is trying to achieve. Together, these files are inputted into a planner such as Fast Downward[1], from which a robot's plan is then generated.

Because writing a domain file can be time intensive and requires the writer to have specialized planning knowledge, standard benchmark instances released by the IPC for general use exist. Figure 3.1 shows a sample action from the Blocksworld PDDL domain, taken from one of these instances.

```
(:action stack
 :parameters (?x - block ?y - block)
 :precondition(and(holding ?x)(clear ?y))
 :effect
(and (not(holding ?x))
(not (clear ?y))
(clear ?x)
(handempty)))
```

Figure 3.1: The stack action as depicted in the IPC 2000 Blocksworld domain. Here, a robot tries to reassemble stackable blocks on a table. To execute the above action, the robot must be holding a block and the top of the block it wants to stack it on, clear. If these positive preconditions are satisfied, the robot can stack the block, resulting in a series of add effects (i.e. the robot's block is stacked on the target block and its top is clear, and the robot's hand is empty) and delete effects (i.e. the robot is no longer holding anything and the target block's top is no longer clear).

[1]https://www.fast-downward.org/

Chapter 4

A GREENHOUSE DISASTER

Picture a robot completing various chores inside a greenhouse to help maintain its operation. Here, a human assigns tasks to the robot based on their beliefs about it. The robot then generates a plan to accomplish these tasks by reconciling its own state/capabilities with the human's directions. In an ideal situation, the robot's execution of this plan would lead it to a goal state perfectly aligned with the human's expectations. However, in a less favorable situation, the robot's execution of this plan may instead cause it to perform actions with potentially dangerous side effects, unanticipated by the human and their belief's about the robot.

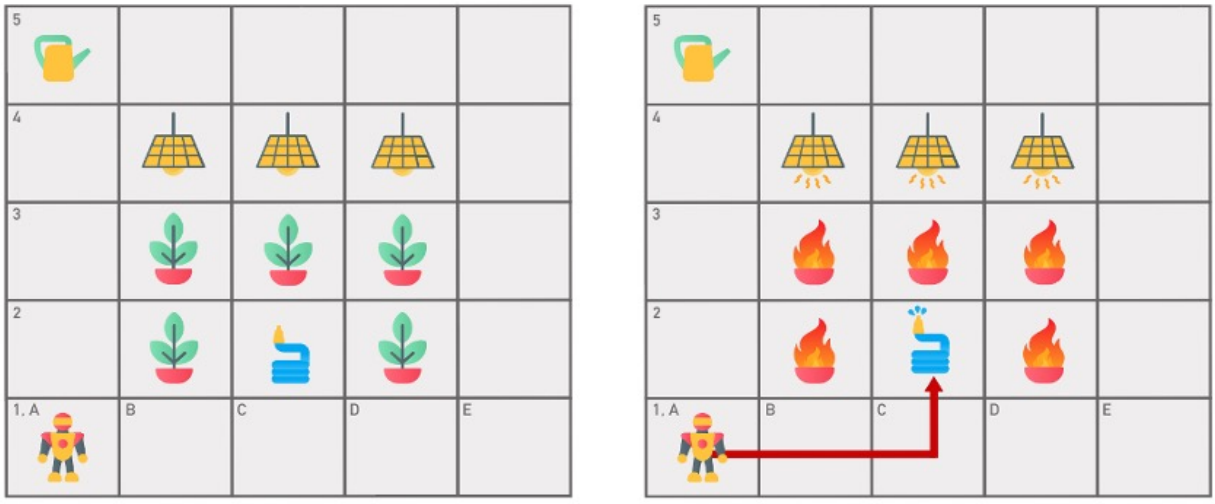


Figure 4.1: In a greenhouse setting, a human asks a robot to water plants based on their incorrect beliefs about its current state/capabilities. To the surprise of the human, the robot chooses to water the plants using a hose, setting the greenhouse on fire after it gets a series of heat lamps wet.

For instance, imagine a scenario where a human asks the robot to water a section of plants, resting beneath a series of heat lamps. Here, the human expects the robot to use a watering pail to complete the task. Instead, the robot surprises the human by grabbing a

nearby hose and haphazardly spraying the plants, as depicted in Figure 4.1. As the robot splashes water all over, it also sprays the above heat lamps, resulting in thermal shock from a sudden change in temperature. Moments later, the heat lamps shatter, releasing sparks and igniting the plants below. In a chain reaction, the greenhouse suddenly catches on fire, as the human – who was unaware the robot could use a hose – looks on in horror.

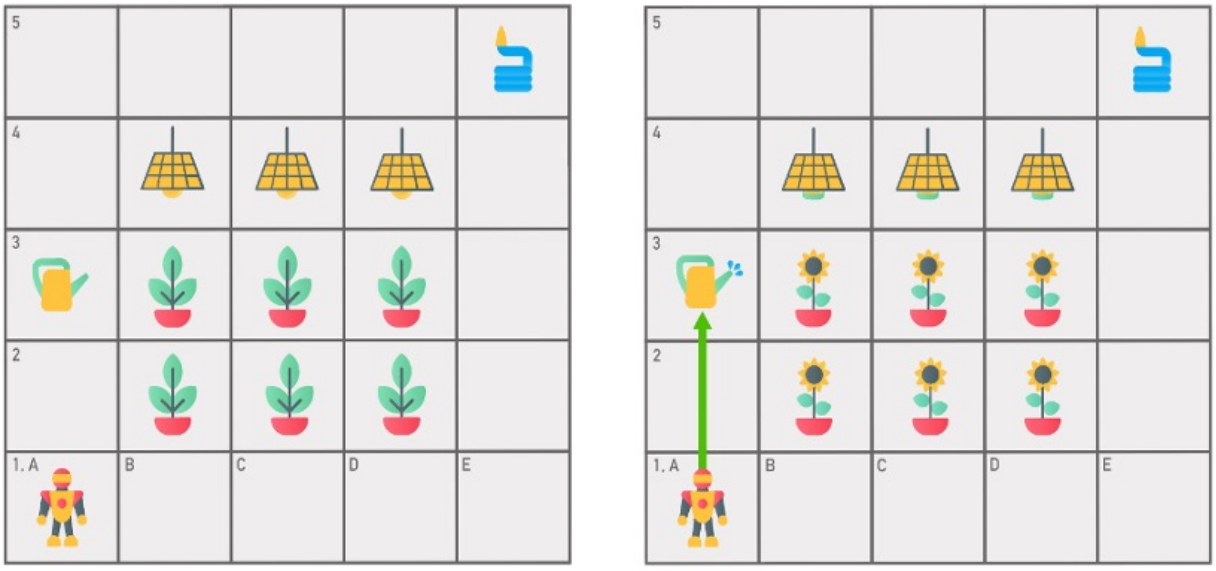


Figure 4.2: To influence which tool a robot selects to water greenhouse plants with, environment design is used to alter their placement. Protective covers are also placed over the greenhouse's lamps to avoid any water exposure. This ensures the plants are properly cared for while mostly avoiding unwanted side effects.

Though extreme, the greenhouse example illustrates how a seemingly benign goal specification, could turn unexpectedly deadly. Further, it highlights the importance of using tools like environment design to influence a robot's behavior, to avoid situations like this. In the greenhouse, for example, the placement of tools could have been altered to influence which one the robot selected for use. Whereas the hose could have been moved farther away from the robot, the watering pail could have been placed closer to it. This change in proximity and increased accessibility (shown in Figure 4.2) would have likely made the watering pail

part of the robot's most optimal plan, increasing the likelihood of its use. For extra safety, protective covers could have also been placed over the heat lamps, preventing any exposure to water at times when the plants were being attended to.

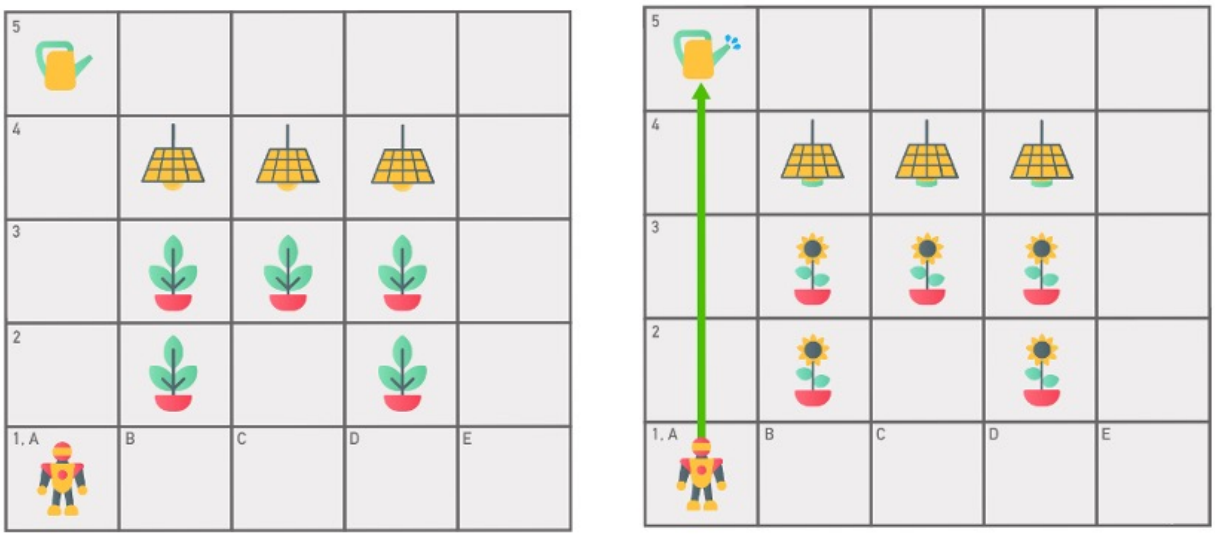


Figure 4.3: To prevent the robot from causing a fire by watering the greenhouse plants with a hose, it is removed from the environment.

However, while these changes would likely result in the robot successfully watering the plants without causing a greenhouse fire – meeting the human's expectations – they wouldn't guarantee it. This is because multiple plans can satisfy a goal state (detailed more in Section 5.2), meaning for as long as the hose was a part of the robot's environment, at least one plan would always incorporate it. Therefore, to avoid any chance of the robot ever using the hose, it would need to be completely removed from the robot's environment or stored in a completely inaccessible way (detailed in Figure 4.3). For this reason, environment design is often a more useful tool in structured settings, where clearer lines can be drawn between the negative side effects caused by a robot's behavior and the environment modifications needed to reliably correct them.

In the rest of this work, I specifically focus on a robot operating in a structured setting and propose an algorithm which can be used to identify and mitigate any undesirable side effects caused by its actions, like the greenhouse fire.

Chapter 5

GOAL STATE DIVERGENCE

As discussed earlier, mismatches between what a human believes a robot's state/capabilities to be and what they realistically are, may cause them to misspecify their goals, potentially resulting in unintended side effects. To prevent this, we now introduce Goal State Divergence ($\mathcal{GSD}$), a metric which measures this expectation gap by comparing a robot's final goal state to a human's expected one (illustrated in Figure 5.1). Through the implementation of this method, our main objectives are that:

1. The robot achieves its goal following the most optimal plan possible.

2. The robot ends in a final goal state closely aligned with a human's expectations.

In this chapter, we first discuss the assumptions underpinning this method. We then ground the concept of $\mathcal{GSD}$ by introducing a series of metrics for quantifying this difference and show how design can be used to minimize it.

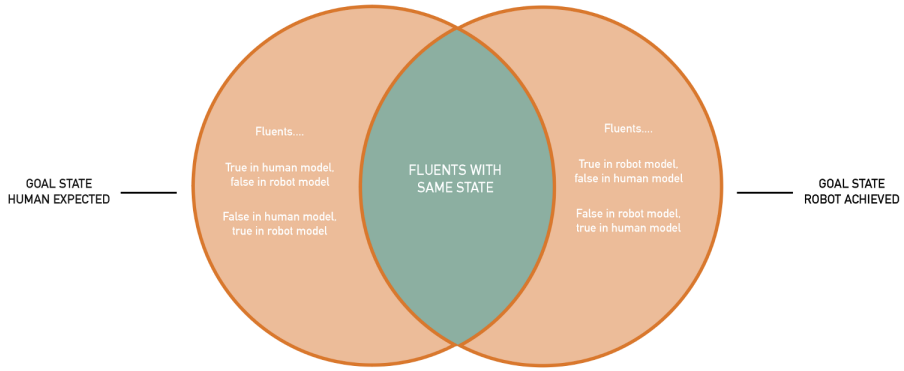


Figure 5.1: Goal State Divergence measures the difference between a human and robot's final goal state fluents.

5.1 Assumptions

Our work begins with the assumption that a human user is assigning a single goal to an autonomous though not necessarily physically embodied robot. This is to avoid confusion and to simplify our problem, rather than an essential requirement for its successful implementation. As such, our work could be easily extended to include different types of agents or have multiple goals assigned to the principal agent.

Central to $\mathcal{GSD}$ is the human's belief model of the robot, which may not match its actual state or capabilities. Because we care about the robot ending in a final goal state closely aligned with a human's expectations, we will therefore assume both sides share the same goal and fluent set. These assumptions are key because they are the model components that will allow us to compare the human and robot's final goal states once both parties have achieved their goal.

So $\mathcal{GSD}$ can be calculated for our setting, we next assume that the robot has access to the human's model and plan. While this may seem like a significant assumption to make, many works have studied learning models in general [44], with several specifically addressing this problem [45]. Further, it is well established that classical planning models correspond to folk psychological models of action and planning used by people. As such, several explainable AI works using classical planning models to capture human beliefs about planning problems exist [46]. We leverage this existing work so the robot can learn the human's model and reuse their domain knowledge for multiple tasks and multiple sets of people when they share the same model [47]. When relevant, this allows the robot to learn one shared human model instead of unnecessarily learning a unique model for each human.

Lastly, we assume the robot is operating in a structured setting. This reduces our problem complexity and on occasion, may result in the human's model already being known. For example, in our greenhouse setting, the human might already have significant experience

working with a previous version of the robot. In turn, their beliefs about what a new version of the robot could do, would have been significantly shaped by the old model's capabilities.

5.2 Design to Reduce Goal State Divergence

To quantify $\mathcal{GSD}$, we begin by comparing pairs of arbitrary states by finding the symmetric difference[1] between them. This involves finding the fluents present in either state but not in the intersection of both states, using a set operation denoted by Δ:

Definition 1. *Given any two states, s^1, s^2, state divergence ($\mathcal{SD}$) is defined as the symmetric difference between their respective fluents, i.e.:*

$$\mathcal{SD}(s^1, s^2) = s^1 \Delta\ s^2$$

Though the symmetric difference between two arbitrary states may reveal some interesting information, this is not the focus of our work. Rather, we care about measuring the difference between a human's expected final goal state and the one achieved by a robot. To accomplish this, we next measure the difference in states across models:

Definition 2. *For a pair of models that are not necessarily distinct, $\mathcal{M}^1$ and $\mathcal{M}^2$, let π^1 be a valid plan in $\mathcal{M}^1$, and π^2 be a valid plan in $\mathcal{M}^2$. Given this, goal state divergence ($\mathcal{GSD}$) of the plan-model pairs is defined as the state divergence between the final state of these two plans, i.e.:*

$$\mathcal{GSD}(\pi^1, \mathcal{M}^1, \pi^2, \mathcal{M}^2) = \mathcal{SD}(\mathcal{T}_{\mathcal{M}^1}(\mathcal{I}^1, \pi^1)\ ,\ \mathcal{T}_{\mathcal{M}^2}(\mathcal{I}^2, \pi^2))$$

[1] https://en.wikipedia.org/wiki/Symmetric_difference

In our setting, these two arbitrary models correspond to the robot's model, $\mathcal{M}^{\mathcal{R}} = \langle \mathcal{D}^{\mathcal{R}}, \mathcal{P}^{\mathcal{R}} \rangle$ and the human's belief model of the robot, $\mathcal{M}^{\mathcal{H}} = \langle \mathcal{D}^{\mathcal{H}}, \mathcal{P}^{\mathcal{H}} \rangle$. In each model, $\mathcal{D}^{\mathcal{R}}$ and $\mathcal{D}^{\mathcal{H}}$ are domain files consisting of fluents, actions, and action costs. Similarly, $\mathcal{P}^{\mathcal{R}}$ and $\mathcal{P}^{\mathcal{H}}$ are problem files, containing objects, initial states, and goal specifications.

We show these more specifically by the robot's model $\mathcal{M}^{\mathcal{R}} = \langle \mathcal{D}^{\mathcal{R}}, \mathcal{I}^{\mathcal{R}}, \mathcal{G}^{\mathcal{R}} \rangle$ and the human's belief model of the robot $\mathcal{M}^{\mathcal{H}} = \langle \mathcal{D}^{\mathcal{H}}, \mathcal{I}^{\mathcal{H}}, \mathcal{G}^{\mathcal{H}} \rangle$. Where $\mathcal{D}^{\mathcal{R}} = \langle \mathcal{F}^{\mathcal{R}}, \mathcal{A}^{\mathcal{R}} \rangle$ and $\mathcal{D}^{\mathcal{H}} = \langle \mathcal{F}^{\mathcal{H}}, \mathcal{A}^{\mathcal{H}} \rangle$. Here, we denote the human and robot's shared goals by $\mathcal{G}^{\mathcal{H}}$ and $\mathcal{G}^{\mathcal{R}}$ (recall $\mathcal{G}^{\mathcal{H}} = \mathcal{G}^{\mathcal{R}}$), the actions present in their models by $\mathcal{A}^{\mathcal{R}}$ and $\mathcal{A}^{\mathcal{H}}$, and their shared fluent set by $\mathcal{F}$ (where $\mathcal{F}^{\mathcal{R}} = \mathcal{F}^{\mathcal{H}} = \mathcal{F}$). Additionally, we show the human's expected plan by $\pi^{\mathcal{H}}$ and the robot's actual plan by $\pi^{\mathcal{R}}$. Given these models and plans, the $\mathcal{GSD}$ metric then becomes $\mathcal{GSD}(\pi^{\mathcal{H}}, \mathcal{M}^{\mathcal{H}}, \pi^{\mathcal{R}}, \mathcal{M}^{\mathcal{R}})$.

5.2.1 Calculating $\mathcal{GSD}$ When Multiple Plans Satisfy a Goal State

While Definition 2 does quantify the difference in final goal states between two models, it is deficient for calculating the $\mathcal{GSD}$ between a human and a robot. This is because despite the robot having access to the human's model, multiple plans can satisfy a goal state, meaning the human's specific plan may be unknown. Additionally, even when known, the robot wouldn't automatically avoid $\mathcal{GSD}$ because the human's plan may be inexecutable in its model. In such a case, the robot would have to find another plan to execute and accept a divergence between its final goal state and the human's despite knowing their model and plan. Given this, we approximate $\mathcal{GSD}$ using an upper and a lower bound, instead of precisely calculating it.

We begin by finding the upper bound or worst-case approximation, denoted by $\mathcal{GD}^{\uparrow}$. This bound corresponds to the maximum divergence possible between a human's expected final goal state and the true goal state achieved by the robot. For example, a scenario where the human expected the robot to achieve a greenhouse of flowers, but instead it achieved

a greenhouse of flames. This means when compared, the largest number of fluents possible will disagree between these two states. We define this by:

Definition 3. *For two given models, $\mathcal{M}^1$, $\mathcal{M}^2$, the worst-case or maximal goal divergence ($\mathcal{GD}^{\uparrow}$) is given by the cardinality of the maximum goal state divergence possible between all executable plans in $\mathcal{M}^1$, $\Pi_{\mathcal{M}^1}$, and $\mathcal{M}^2$, $\Pi_{\mathcal{M}^2}$, i.e.:*

$$\mathcal{GD}^{\uparrow}(\mathcal{M}^1, \mathcal{M}^2) = \max_{\pi^1 \in \Pi_{\mathcal{M}^1}, \pi^2 \in \Pi_{\mathcal{M}^2}} (|\mathcal{GSD}(\pi^1, \mathcal{M}^1, \pi^2, \mathcal{M}^2)|)$$

This definition is followed by Proposition 1, which asserts that the $\mathcal{GD}^{\uparrow}$ will always be an upper bound of the true $\mathcal{GSD}$.

Proposition 1. *For the robot and human model pair $\mathcal{M}^{\mathcal{R}}$ and $\mathcal{M}^{\mathcal{H}}$, the maximal goal divergence is guaranteed to be greater than or equal to the goal state divergence for the human plan $\pi^{\mathcal{H}}$ and the robot plan $\pi^{\mathcal{R}}$, i.e., $\mathcal{GD}^{\uparrow}(\mathcal{M}^{\mathcal{H}}, \mathcal{M}^{\mathcal{R}}) \geq |\mathcal{GSD}(\pi^{\mathcal{H}}, \mathcal{M}^{\mathcal{H}}, \pi^{\mathcal{R}}, \mathcal{M}^{\mathcal{R}})|$.*

Using Definition 3, this proposition can be easily proven and used to reduce $\mathcal{GSD}$ by simply decreasing the $\mathcal{GD}^{\uparrow}$. This is especially relevant in cases where the $\mathcal{GD}^{\uparrow}$ is reducible to zero because this reduction would guarantee upper and lower $\mathcal{GSD}$ bounds of zero, meaning perfect alignment between the human and robot's final goal states (i.e. an empty set). However, on occasion the $\mathcal{GD}^{\uparrow}$ may include plans that will likely never be considered by the human or executed by the robot. In situations like this, where the $\mathcal{GD}^{\uparrow}$ is a loose upper bound, minimizing this value won't always reduce $\mathcal{GSD}$. For this reason, we look to instead reduce $\mathcal{GSD}$ by approximating its lower bound or the best-case approximation, denoted by $\mathcal{GD}^{\downarrow}$. We illustrate this in Figure 5.2.

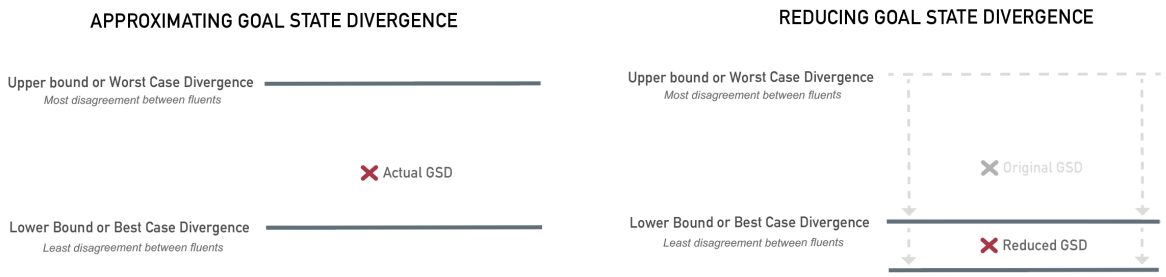


Figure 5.2: In most problem settings, only reducing $\mathcal{GD}^{\uparrow}$ won't lower the true $\mathcal{GSD}$ too. This means to find a close $\mathcal{GSD}$ approximation, reducing the $\mathcal{GD}^{\uparrow}$ and the $\mathcal{GD}^{\downarrow}$ is key.

Unlike the $\mathcal{GD}^{\uparrow}$, the $\mathcal{GD}^{\downarrow}$ corresponds to the smallest divergence possible between a human's expected final goal state and the true goal state achieved by a robot. For example, a scenario where the human expected the robot to achieve a greenhouse of flowers and it did, but left the watering pail in an unexpected spot. This means when compared, the largest number of fluents possible will agree between these two states. It is important to note that this number corresponds to the cardinality of the $\mathcal{GD}^{\downarrow}$, meaning for this measure, the human and robot's final goal states will either match exactly or have their divergence captured by a non-negative number. We define this by:

Definition 4. *For two given models, $\mathcal{M}^1$, $\mathcal{M}^2$, the best-case or minimal goal divergence ($\mathcal{GD}^{\downarrow}$) is given by the cardinality of the minimum goal state divergence possible between all executable plans in $\mathcal{M}^1$, $\Pi_{\mathcal{M}^1}$, and $\mathcal{M}^2$, $\Pi_{\mathcal{M}^2}$, i.e.:*

$$\mathcal{GD}^{\downarrow}(\mathcal{M}^1, \mathcal{M}^2) = \min_{\pi^1 \in \Pi_{\mathcal{M}^1}, \pi^2 \in \Pi_{\mathcal{M}^2}} (|\mathcal{GSD}(\pi^1, \mathcal{M}^1, \pi^2, \mathcal{M}^2)|)$$

As before, we follow this definition with a proposition, which maintains that the $\mathcal{GD}^{\downarrow}$ is a lower bound because it will always be smaller than the true $\mathcal{GSD}$ value.

Proposition 2. *For the robot and human model pair $\mathcal{M}^{\mathcal{R}}$ and $\mathcal{M}^{\mathcal{H}}$, the minimal goal divergence is guaranteed to be less than or equal to the goal state divergence for the human plan $\pi^{\mathcal{H}}$ and the robot plan $\pi^{\mathcal{R}}$, i.e., $\mathcal{GD}^{\downarrow}(\mathcal{M}^{\mathcal{H}}, \mathcal{M}^{\mathcal{R}}) \leq |\mathcal{GSD}(\pi^{\mathcal{H}}, \mathcal{M}^{\mathcal{H}}, \pi^{\mathcal{R}}, \mathcal{M}^{\mathcal{R}})|$.*

Using the gap between our upper and lower bounds, we can now approximate where in the space between them $\mathcal{GSD}$ exists:

$$\mathcal{GSD} \approx \mathcal{GD}^{\uparrow}(\mathcal{M}^{\mathcal{H}}, \mathcal{M}^{\mathcal{R}}) - \mathcal{GD}^{\downarrow}(\mathcal{M}^{\mathcal{H}}, \mathcal{M}^{\mathcal{R}})$$

Here, we are essentially finding a confidence interval indicating how accurate our $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$ approximations are. Whereas a smaller difference between these bounds would reflect a more confident estimation, a larger difference between them would suggest a more uncertain one. The only exception to this being a $\mathcal{GSD}$ of zero, which would be an exact value because it would mean the human and robot's final goal states match exactly.

5.2.2 Guaranteeing Unwanted Side Effects are Prevented

While this may help in characterizing $\mathcal{GSD}$ for a given problem, it's important to note that the above definitions are effectively considering the space of all plans. This means weaker, less effective bounds, where a space of models much larger than what the human or robot would ever expect or be able to execute, is being considered. One consequence of this, is the robot may execute completely sub-optimal plans, unexpected by the human. For example, when instructed to water greenhouse flowers, instead of following a direct path to the plants, the robot may instead complete ten circle revolutions first. Another consequence of this is the inclusion of plans where the robot will inevitably perform actions with undesirable side effects. For instance, in the greenhouse, the hose may consistently

be a part of one or more of the robot's plans, regardless of its placement. This means the greenhouse would forever be at risk of burning down because the robot would always have a plan where it sprays the plants with the hose, causing the heat lamps to shatter. Here, the only way to prevent this outcome would be to either entirely remove the hose from the robot's environment or leave it in an inaccessible state, preventing its use.

In this work, we purely focus on the bounds calculated over a set of plans, which we don't limit to only those the human or robot are most likely to execute. However, in the appendix of this document, we provide a detailed look at how our methods could be extended to account for settings like this.

5.2.3 Environment Modifications & The Designer

Now, having established the concept of $\mathcal{GSD}$ and metrics that quantify it, we will discuss how design can be used to reduce this measure. Specifically, we will make modifications to the robot's model by adding or removing specific elements from it, pushing it towards a final goal state more aligned with a human's expectations. We begin by defining our design problem:

Definition 5. *A **human-robot goal alignment design (HRGAD)** problem is characterized by the tuple, $\mathcal{DP} = \langle \mathcal{M}^{\mathcal{R}}, \mathcal{M}^{\mathcal{H}}, \mathbb{U}, \Lambda, \mathcal{C} \rangle$, where:*

- $\mathcal{M}^{\mathcal{R}}$, $\mathcal{M}^{\mathcal{H}}$, *are the initial robot and human models.*
- $\mathbb{U}$ *is a set of available environment modifications*[2] *or model updates. These may include changes to the state space, action preconditions, action effects, action costs, initial state, or goal.*

[2]Throughout this work, the terms modification, design modification and environment modification are used interchangeably.

- $\Lambda : \mathbb{M} \times \mathbb{U} \rightarrow \mathbb{M}$ *is the transition function over a space of possible models. The function generates the model that would be obtained by performing the set of modifications on a given model.*
- $\mathcal{C}$ *is an additive cost function that maps each design modification in* $\mathbb{U}$ *to a cost.*

Given our greenhouse, a model update could take a variety of forms, including moving the plants around inside it, putting protective covers over the heat lamps or introducing a new watering tool to it. Here, such modifications are made offline by an outside agent called the designer, prior to the human or robot becoming active in the environment. This allows the designer to proactively address any known problems the robot's behavior may cause. Though such modifications are intended as more of a fail-safe, rather than a fully preventative measure, and are selected based on the assumption that the human and robot's behavior can be modeled and predicted. By having the designer implement them, the robot is then spared from having to query the human for information or reason about their beliefs, which previous work has shown to be a much more cost-effective way of enforcing desirable behavior [40]. Further, this exempts the robot and human from having to make design modifications on their own.

Using the above metrics, many different classes of solutions for this problem could be defined. For the remainder of this work, the solution we concentrate on is minimizing the design cost of this problem, while requiring the upper and lower $\mathcal{GSD}$ bounds to dip below a certain threshold. We do this by finding a minimal set of design modifications that when applied to a model, achieve this. We more formally define this by:

Definition 6. *An* (l, u)*-bounded minimal solution to a* $\mathcal{DP}$ *is a set of modifications* [3] ξ^* *that minimizes the following constrained optimization problem.*

[3]This definition makes an implicit assumption that each design is independent and, as such, can be performed in any order.

$$\min_{\xi \in 2^{\mathbb{U}}} \mathcal{C}(\xi)$$

Such that $\mathcal{GD}^{\downarrow}(\mathcal{M}_{\xi}^{\mathcal{R}}, \mathcal{M}_{\xi}^{\mathcal{H}}) \leq \ell$ *and* $\mathcal{GD}^{\uparrow}(\mathcal{M}_{\xi}^{\mathcal{R}}, \mathcal{M}_{\xi}^{\mathcal{H}}) \leq u$, *where* $\mathcal{M}_{\xi}^{\mathcal{R}} = \Lambda(\mathcal{M}^{\mathcal{R}}, \xi)$ *and* $\mathcal{M}_{\xi}^{\mathcal{H}} = \Lambda(\mathcal{M}^{\mathcal{H}}, \xi)$

Recall, in our setting, the robot must only achieve a single goal. For this unique goal specification, we implement the above approach by only allowing designs that effect the robot's initial state to be made. This is relevant because outside of their initial states, the human and robot both share the same exact fluents, goal specification and action preconditions, effects, and costs. This means the human and robot's initial states are the primary elements driving what plan either side is selecting to achieve their goals. Thus, by only modifying the their initial states, we can perform a more controlled set of experiments where the optimal upper and lower bounds to reduce $\mathcal{GSD}$ for each instance becomes clear. However, for settings where a robot must achieve multiple goals each with a unique specification, Definition 6 could be easily extended and the number and type of design modifications available to make broadened.

In the evaluation section of this work, we find the $\mathcal{GD}^{\uparrow}$ and the $\mathcal{GD}^{\downarrow}$ values across instances for our single problem setting. However, in alternate settings, these metrics might vary. For example, if the problem were modified to incorporate different designs to minimize the worst-case divergence across all instances, ensuring the max of the $\mathcal{GD}^{\uparrow}$ and the $\mathcal{GD}^{\downarrow}$ was below a certain threshold would be important. In situations where a robot might be trying to achieve one goal out of a set of possible goals, analyzing how designs associated with different tasks might change the $\mathcal{GD}^{\downarrow}$ and $\mathcal{GD}^{\uparrow}$ would be important. In such a case, the solutions would be nearly identical to those discussed in the next section.

5.3 Calculating $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$

Now with a basic understanding of our design problem set, we turn our attention to calculating approximations of $\mathcal{GSD}$. Using an off-the-shelf cost optimal planner, we create a single planning problem by formulating actions in the human and robot's models, each operating on a different copy of the state fluents. Using this compilation, a plan is generated to achieve the specified goal for both models. Check actions are then used to compare the final goal state achieved by the human's plan and model to the one achieved by the robot's plan and model. To identify the upper and lower $\mathcal{GSD}$ bounds, we will modulate our costs to encourage the planner to identify plans whose final goal states have more or less overlap. A diagrammatic representation of the compilation is shown in Figure 5.3. In this section, we present the syntax associated with this novel compilation and a detailed overview of how it works.

5.3.1 Model Compilation

Model compilation is a technique for simplifying a planning problem so it can be used more easily by a planner. For our problem, we do this by transforming the model pair $\lambda = \langle \mathcal{M}^{\mathcal{R}}, \mathcal{M}^{\mathcal{H}} \rangle$ into a new compiled model where $\mathcal{M}^{\lambda} = \langle \mathcal{D}^{\lambda}, \mathcal{I}^{\lambda}, \mathcal{G}^{\lambda} \rangle$. Like before, $\mathcal{D}^{\lambda}$ is a domain containing fluents and actions, $\mathcal{I}^{\lambda}$ some initial state, and $\mathcal{G}^{\lambda}$ a goal specification. However, these components now contain new and additional elements. While $\mathcal{D}^{\lambda}$ remains defined by the tuple $\mathcal{D}^{\lambda} = \langle \mathcal{F}^{\lambda}, \mathcal{A}^{\lambda} \rangle$, its contents have been expanded as sets, with $\mathcal{F}^{\lambda}$ now containing additional fluents and $\mathcal{A}^{\lambda}$ more actions.

Accordingly, $\mathcal{F}^{\lambda}$ is now shown by $\mathcal{F}^{\lambda} = \mathcal{F}^{\mathcal{R}} \cup \mathcal{F}^{\mathcal{H}} \cup \mathcal{F}^{\theta} \cup \mathcal{F}^{\kappa}$. Here, $\mathcal{F}^{\mathcal{R}}$ corresponds to the original set of robot fluents, which we individually denote by $f_i^{\mathcal{R}}$ (i.e. $f_i^{\mathcal{R}} \in \mathcal{F}^{\mathcal{R}}$). These are used to keep track of how the robot's plan and final goal state unfolds. To illustrate this, imagine there is a set of greenhouse plants in the robot's model that need to

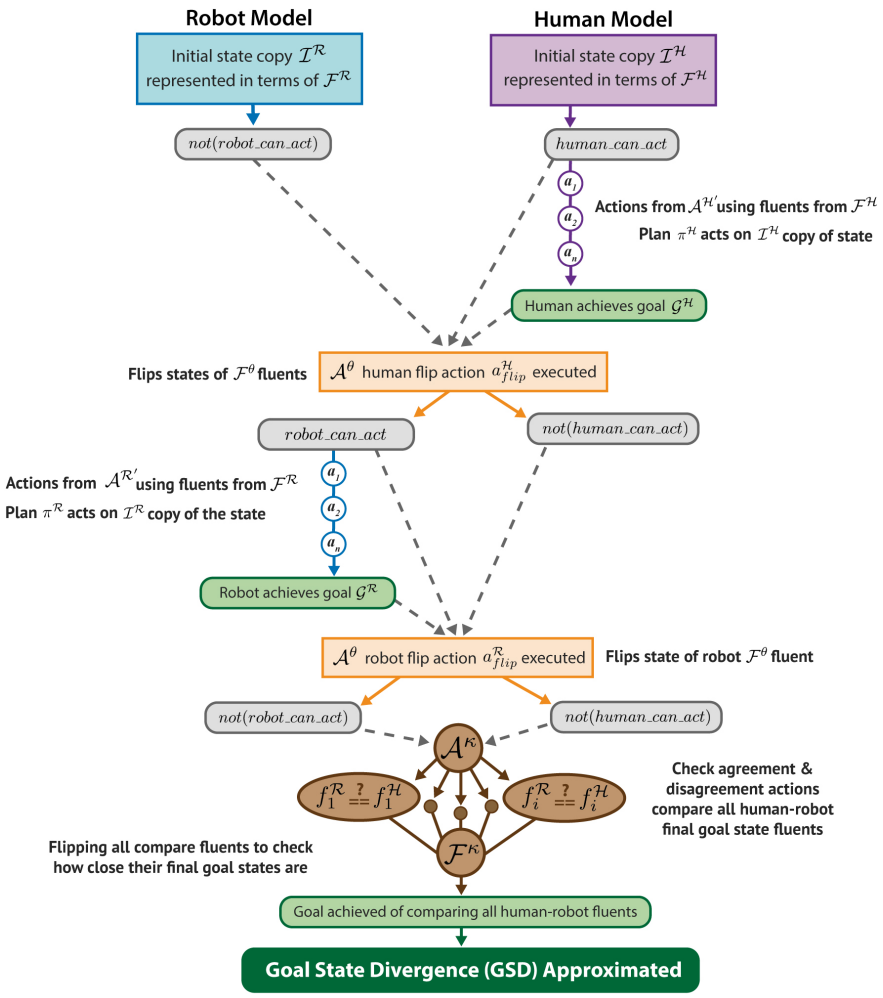


Figure 5.3: An illustration of the $\mathcal{GSD}$ Compilation.

be watered. Using the fluent `(robot_plants_watered)`, we could check if these plants were indeed watered at the end of the robot's plan (i.e. if `(robot_plants_watered)` is true).

Similarly, $\mathcal{F}^{\mathcal{H}}$ corresponds to the human's set of belief fluents about the robot, which we individually denote by $f^{\mathcal{H}}_i$. These fluents are an exact copy of the robot's fluents in $\mathcal{F}^{\mathcal{R}}$, meaning for every $f^{\mathcal{R}}_i \in \mathcal{F}^{\mathcal{R}}$, a $f^{\mathcal{H}}_i$ copy fluent exists. These are used to keep track of how the human believes the robot's plan and final goal state will unfold, relative to what they believe its current state and capabilities to be. Like before, imagine there are plants in the human's greenhouse that need to be watered, identical to those in the robot's greenhouse. Using the

fluent `(human_copy_plants_watered)`, we could check if at the end of the human's plan, these plants were watered too (i.e. if `(human_copy_plants_watered)` is true).

$\mathcal{F}^{\theta}$ represents the special housekeeping fluents $\{human\_can\_act\}$ and $\{robot\_can\_act\}$, which are used to control when the human or robot can perform actions. Though not a requirement, this allows us to enforce an ordering between the human and robot's actions for a more efficient compilation.[4] Additionally, these fluents are also used to prevent the human and robot from performing any additional actions once their final goal states have been achieved. This ensures these states remain unchanged prior to their $\mathcal{GSD}$ being calculated, guaranteeing that the estimation returned by the planner is correct.

$\mathcal{F}^{\kappa}$ corresponds to compare fluents, which we individually denote by f_i^{κ} and for which one exists for every $f_i^{\mathcal{R}}$ (i.e. for every $f_i^{\mathcal{R}} \in \mathcal{F}^{\mathcal{R}}, \exists f_i^{\kappa} \in \mathcal{F}^{\kappa}$). We use these to track whether corresponding pairs of fluents from the human and robot's final goal states have been compared. This process sits at the heart of the $\mathcal{GSD}$ metric, which we approximate based on whether or not these fluents match. Here, `(robot_plants_watered)` and `(human_copy_plants_watered)` would be an example corresponding fluent pair. Using a compare fluent like `(compared_plants_watered)`, we could track if the final state of all fluent pairs like this have been compared (i.e. `(compared_plants_watered)` is true).

Next in our compiled model is the initial state $\mathcal{I}^{\lambda}$, now defined by $\mathcal{I}^{\lambda} = \mathcal{I}^{\mathcal{R}} \cup \mathcal{I}^{\mathcal{H}} \cup \{human\_can\_act\}$. Here, $\mathcal{I}^{\mathcal{R}}$ represents the robot's initial state. This is the true version of the world the robot begins operating in at the start of its planning process. In the greenhouse, this may include where certain plants are located, what tools are available to water them with, and what the interior layout of the greenhouse is like. $\mathcal{I}^{\mathcal{H}}$ is then the human's beliefs about the robot's initial state, which is the starting conditions they think the robot is beginning its planning process with. These beliefs may not match true reality,

[4]We discuss this constraint more in our evaluation section, where multiple compilation versions are tested, some with the actions enforced and some without.

in which case plants in the human's version of the greenhouse may be in different positions, certain tools missing, and its layout completely changed. We include the special housekeeping fluent $\{human\_can\_act\}$ as part of the initial state because for our problem, the human begins in a state able to perform actions, whereas the robot does not.[5]

$\mathcal{G}^\lambda$ represents the human and robot's shared goals, now defined by $\mathcal{G}^\lambda = \mathcal{G}^\mathcal{R} \cup \mathcal{G}^\mathcal{H} \cup \mathcal{F}^\kappa$. Within this set, $\mathcal{G}^\mathcal{R}$ and $\mathcal{G}^\mathcal{H}$ correspond to the robot and human's goal specification, each expressed using a subset of their original or copy fluents (i.e. $\mathcal{G}^\mathcal{R} \subseteq \mathcal{F}^\mathcal{R}$, $\mathcal{G}^\mathcal{H} \subseteq \mathcal{F}^\mathcal{H}$). This is because while many fluents may describe a state, far fewer are needed to describe a goal. As mentioned earlier, because the objective of this compilation is to compare the human and robot's final goal states, these goal specifications are the same (i.e. $\mathcal{G}^\mathcal{H} = \mathcal{G}^\mathcal{R}$). Meaning, the human and robot are both trying to achieve the same goal, like watering all plants inside their respective greenhouses. Also included in this set, to compare the human and robot's final goal states once all goals have been achieved, is the original check fluents, $\mathcal{F}^\kappa$. This means to calculate $\mathcal{GSD}$, the human and robot must have both achieved their goals (e.g. `(robot_plants_watered)` and `(human_copy_plants_watered)`) and one compare fluent for every original fluent in the robot's model must have been compared (e.g. `(compared_plants_watered)`).

Last in our compiled model is $\mathcal{A}^\lambda$, which relates to the actions the robot can perform, as well as those the human believes it can perform. We represent this as an updated set, denoted by $\mathcal{A}^\lambda = \mathcal{A}^{\mathcal{R}'} \cup \mathcal{A}^{\mathcal{H}'} \cup \mathcal{A}^\theta \cup \mathcal{A}^\kappa$. Here, $\mathcal{A}^{\mathcal{R}'}$ represents the set of actions executable in the robot's model, which are nearly identical to those in $\mathcal{A}^\mathcal{R}$. The key difference being that all actions in $\mathcal{A}^{\mathcal{R}'}$ contain the precondition $\{robot\_can\_act\}$, dictating when they can be performed. These actions contain fluents from $\mathcal{F}^\mathcal{R}$.

[5]However, if the $\{robot\_can\_act\}$ fluent was added to the initial state, this enforced ordering would be removed and the human/robot could both simultaneously perform actions.

$\mathcal{A}^{\mathcal{H}'}$ corresponds to the human's belief set of actions and are nearly identical to all definitions in $\mathcal{A}^{\mathcal{H}}$, except each contains a $\{human\_can\_act\}$ precondition. These contain fluents from $\mathcal{F}^{\mathcal{H}}$ and represent the actions the human thinks the robot can perform but which may not match true reality. This means there may be some actions in the human's model that are missing in the robot's model because it doesn't possess the proper state or capabilities to execute them. Conversely, there may be some actions in the robot's model, that are missing in the human's model because they don't believe the robot can execute them. For example, in the greenhouse, because the human is not aware the robot can use a hose, the action *use_hose* action would be missing from their model but included in the robot's. It is important to note that for our specific problem, the actions present in the human and robot's models are identical. However, in an alternate setting, this may not be the case.

$\mathcal{A}^{\theta}$ are special flip actions that enable or disable the robot or human's ability to perform actions by changing the state of the $\mathcal{F}^{\theta}$ fluents (i.e. changing $\{robot\_can\_act\}$ or $\{human\_can\_act\}$ to true or false). These actions are denoted by $a^{\mathcal{H}}_{flip}$ and $a^{\mathcal{R}}_{flip}$, and important because they guarantee that a valid plan has been found in the human and robot's models before any final goal state fluents have been compared. These actions are also important because as mentioned with the $\mathcal{F}^{\theta}$ fluents, they prevent the human and robot from performing any additional actions once their final goal states have been achieved, ensuring the planner can correctly estimate their $\mathcal{GSD}$. In this work, we assume all actions in $\mathcal{A}^{\theta}$ have a unit cost of zero. We define the human's flip action followed by the robot's flip action as follows:

- $pre_{+}(a^{\mathcal{H}}_{flip})/\{human\_can\_act\} \subseteq \mathcal{G}^{\mathcal{H}}$,
  $pre_{-}(a^{\mathcal{H}}_{flip}) = \emptyset$:

$add(a^{\mathcal{H}}_{flip}) = \{robot\_can\_act\}$,

$del(a^{\mathcal{H}}_{flip}) = \{human\_can\_act\}$

- $pre_{+}(a^{\mathcal{R}}_{flip})/\{robot\_can\_act\} \subseteq \mathcal{G}^{\mathcal{R}}$,
  $pre_{-}(a^{\mathcal{R}}_{flip}) = \emptyset$:
  $add(a^{\mathcal{R}}_{flip}) = \emptyset$,
  $del(a^{\mathcal{R}}_{flip}) = \{robot\_can\_act\}$

Once the human and robot have both executed their plans, the fluents from their final goal states are compared using a set of compare actions denoted by $\mathcal{A}^{\kappa} = A^{\kappa}_{f_1} \cup A^{\kappa}_{f_2} \cup A^{\kappa}_{f_3} ... A^{\kappa}_{f_{|\mathcal{F}|}}$. Here, for every $f^{\mathcal{R}}_i \in \mathcal{F}^{\mathcal{R}}$, a set of compare actions exists such that $A^{\kappa}_{f_i} = \{a^1_{f_i}, a^2_{f_i}, a^3_{f_i}, a^4_{f_i}\}$. These actions contain $f^{\mathcal{R}}_i$ and $f^{\mathcal{H}}_i$ fluents that are being compared (i.e. `(human_copy_plants_watered)` and `(robot_plants_watered)`), as well as f^{κ}_i fluents for checking if this comparison has been made (i.e. `(compared_plants_watered)`). While corresponding fluents may describe the same attributes of a domain, when compared, their respective states may be the same or completely different as exemplified in Table 5.1.

Scenario	**Robot Fluent**	**Human Fluent**	**Greenhouse Example**
1	True	False	The robot watered its plants but the human didn't water theirs.
2	False	True	The robot didn't water its plants but the human watered theirs.
3	True	True	The robot and human both watered their plants.
4	False	False	Neither the robot nor the human watered their plants.

Table 5.1: Here we show all fluent state combinations, each with an example taken from our greenhouse setting.

To account for these differences unique check agreement and check disagreement compare actions are used to analyze whether a pair of corresponding fluents share the same state. These actions only execute when certain preconditions are met and contain compare fluents for verifying that these comparisons have taken place. Upon execution, each unique check agreement and check disagreement compare action is assigned a cost, which we represent using the symbols $\mathcal{P}_1$ (agreement action cost) and $\mathcal{P}_2$ (disagreement action cost). Based on which $\mathcal{GSD}$ bound we are trying to find, these costs are modulated to encourage the planner to select plans with more agreement or more disagreement between the human and robot's final goal states.

We begin with the check disagreement compare actions, which we denote by $\mathcal{A}^{\kappa-}$. These correlate to the first and second greenhouse scenarios in Table 5.1. These actions only execute when the human and robot are both in states unable to act, and share corresponding fluents with opposite states, never compared before (i.e. $f_i^{\mathcal{R}}$ must be true when $f_i^{\mathcal{H}}$ is false, or $f_i^{\mathcal{H}}$ must be true when $f_i^{\mathcal{R}}$ is false). When trying to find the $\mathcal{GD}^{\uparrow}$, we assign zero cost to these actions to encourage the planner to select human-robot plans with high levels of disagreement. Conversely, when trying to find the $\mathcal{GD}^{\downarrow}$, we assign a steep cost to these actions to encourage the planner to select human-robot plans with low levels of disagreement. We show these check disagreement compare actions by:

- $pre_+(a^1_{f_i}) = \{f_i^{\mathcal{R}}\}$,
  $pre_-(a^1_{f_i}) = \{f_i^{\mathcal{H}}\} \cup \{robot\_can\_act\} \cup \{human\_can\_act\} \cup \{f_i^{\kappa}\}$,
  $add(a^1_{f_i}) = \{f_i^{\kappa}\}$, $del(a^1_{f_i}) = \emptyset$, and $c(a^1_{f_i}) = \mathcal{P}_1$

- $pre_+(a^2_{f_i}) = \{f_i^{\mathcal{H}}\}$,
  $pre_-(a^2_{f_i}) = \{f_i^{\mathcal{R}}\} \cup \{robot\_can\_act\} \cup \{human\_can\_act\} \cup \{f_i^{\kappa}\}$,
  $add(a^2_{f_i}) = \{f_i^{\kappa}\}$, $del(a^2_{f_i}) = \emptyset$, and $c(a^2_{f_i}) = \mathcal{P}_1$

Next is the check agreement compare actions, which we denote by $\mathcal{A}^{\kappa+}$. These actions correlate to the third and fourth greenhouse scenarios in Table 5.1. These actions only execute when the human and robot are both in states unable to act, and share corresponding fluents with matching states, never compared before (i.e. $f_i^{\mathcal{R}}$ and $f_i^{\mathcal{H}}$ must both be true, or $f_i^{\mathcal{R}}$ and $f_i^{\mathcal{H}}$ must both be false). When trying to find the $\mathcal{GD}^{\uparrow}$, we assign a steep cost to these actions to encourage the planner to select human-robot plans with low levels of agreement. Conversely, when trying to find the $\mathcal{GD}^{\downarrow}$, we assign zero cost to these actions to encourage the planner to select human-robot plans with high levels of agreement. We show these check agreement compare actions by:

- $pre_+(a^3_{f_i}) = \{f_i^{\mathcal{R}}, f_i^{\mathcal{H}}\}$,
  $pre_-(a^3_{f_i}) = \{robot\_can\_act\} \cup \{human\_can\_act\} \cup \{f_i^{\kappa}\}$,
  $add(a^3_{f_i}) = \{f_i^{\kappa}\}$, $del(a^3_{f_i}) = \emptyset$, and $c(a^3_{f_i}) = \mathcal{P}_2$

- $pre_+(a^4_{f_i}) = \emptyset$,
  $pre_-(a^4_{f_i}) = \{f_i^{\mathcal{R}}, f_i^{\mathcal{H}}\} \cup \{robot\_can\_act\} \cup \{human\_can\_act\} \cup \{f_i^{\kappa}\}$,
  $add(a^4_{f_i}) = \{f_i^{\kappa}\}$, $del(a^4_{f_i}) = \emptyset$, and $c(a^4_{f_i}) = \mathcal{P}_2$

To approximate our bounds using these compare actions, we start by identifying a new valid plan, π^{λ}, for our new compiled model $\mathcal{M}^{\lambda}$. Comprising this is a sequence of human and robot actions, denoted by $\mathcal{H}(\pi^{\lambda})$ and $\mathcal{R}(\pi^{\lambda})$, respectively. To keep track of the check disagreement and check agreement compare actions appearing in this plan, we use the notation $\kappa^{+}(\pi^{\lambda})$ and $\kappa^{-}(\pi^{\lambda})$. Based on which bound we are trying to find, we assign varying costs to these, forcing the planner to generate the best and worse-case scenario action sequences. This brings us to Proposition 3, where we modulate our cost to produce a plan with the most disagreement:

Proposition 3. *For a given compiled model $\mathcal{M}^{\lambda}$, let us set the action costs of all actions in $\mathcal{A}^{\mathcal{R}'} \cup \mathcal{A}^{\mathcal{H}'}$ to a unit cost, and set the disagreement cost as $\mathcal{P}_2 = 0$ and agreement cost as $\mathcal{P}_1 > 2^{|\mathcal{F}^{\mathcal{R}}|+|\mathcal{F}^{\mathcal{H}}|}$. For the given cost function, let π^{λ} be an optimal plan, then $\mathcal{GD}^{\uparrow}(\mathcal{M}^{\mathcal{R}}, \mathcal{M}^{\mathcal{H}}) = |\kappa^{-}(\pi^{\lambda})|$.*

Here, when a pair of human-robot fluents don't share the same state, a unit cost of zero is assigned. In contrast, when they do match, a cost greater than the combined cost of the longest possible plan in either the human or robot model is assigned. In turn, plans with the greatest disagreement are generated by the planner. To find plans with the most agreement, we invert these costs:

Proposition 4. *For a given compiled model $\mathcal{M}^{\lambda}$, let us set the action costs of all actions in $\mathcal{A}^{\mathcal{R}'} \cup \mathcal{A}^{\mathcal{H}'}$ to a unit cost, and set the agreement cost as $\mathcal{P}_1 = 0$ and disagreement cost as $\mathcal{P}_2 > 2^{|\mathcal{F}^{\mathcal{R}}|+|\mathcal{F}^{\mathcal{H}}|}$. For the given cost function, let π^{λ} be an optimal plan, then $\mathcal{GD}^{\downarrow}(\mathcal{M}^{\mathcal{R}}, \mathcal{M}^{\mathcal{H}}) = |\kappa^{-}(\pi^{\lambda})|$.*

The intuition behind Proposition 4 is identical to Proposition 3, obviating the need for further explanation.

5.4 Identifying Minimal Designs for HRGAD

Now with the methods in place for calculating our $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$ bounds, we turn our attention to identifying designs that will meet the criteria for a Human-Robot Goal Alignment design (HRGAD) problem (detailed in Definitions 5). Recall, that in our original compilation, we approximated $\mathcal{GSD}$ in three steps:

1. Using enforced ordering, the human executed a plan and achieved their goals, followed by the robot.

2. The human and robot's ability to perform actions was turned off.

3. The human and robot's final goal states were compared for consistency.

Now, we slightly modify and extend this compilation, by adding a design phase before this goal achievement phase to reduce, then approximate $\mathcal{GSD}$. Using this new compilation, we consider a specific instantiation of Definition 6, where we only consider settings with a lower bound of zero, and a design set limited to initial state changes, each assigned a unit cost. To find this minimal set of design modifications, we employ a basic algorithm with an outer and an inner loop. We begin in the outer loop, where we arbitrarily set a design budget shown by τ, which iteratively increases. Given this budget constraint, the inner loop tries to identify a design where the $\mathcal{GD}^{\downarrow}$ is zero, and the $\mathcal{GD}^{\uparrow}$ is within some specified limit u. In this section, we detail the specifics of this inner loop, along with an extension of our previously compiled model, to show how these designs may be identified to reduce $\mathcal{GSD}$.

5.4.1 Inner Loop for Identifying Designs

In the inner loop, we begin in the design phase, where the $\mathcal{GD}^{\downarrow}$ compilation is slightly modified to perform design actions corresponding to initial state changes. Using this updated compilation, we find a set of modifications $\mathbb{U}$, that will lower our $\mathcal{GD}^{\downarrow}$ and use our design actions to implement them. Once all changes have been made, we disable these design actions and find a pair of human-robot plans with a $\mathcal{GD}^{\downarrow}$ of zero, by directly encoding into our goal that we are only looking for plans where all final goal state fluents match.

Once zero $\mathcal{GD}^{\downarrow}$ has been achieved, we enter the goal achievement phase. Here we input our set of design modifications into the $\mathcal{GD}^{\uparrow}$ part of the inner loop, and check if they fall within the required u-upper-bound. If they do, this set of design modifications is returned as the final set. The human and robot then execute their plans again, this time achieving a reduced, though not necessarily zero $\mathcal{GD}^{\uparrow}$. If they don't, the compilation is updated

to disallow this set of design modifications and we return to the design phase. Here, the updated $\mathcal{GD}^{\downarrow}$ tries to find another set of designs which are within budget and meet the $\mathcal{GD}^{\downarrow} = 0$ requirement. If no designs can be found, the budget τ is increased, and the search to find another set that fits this criteria continued. We show this process in Figure 5.4.

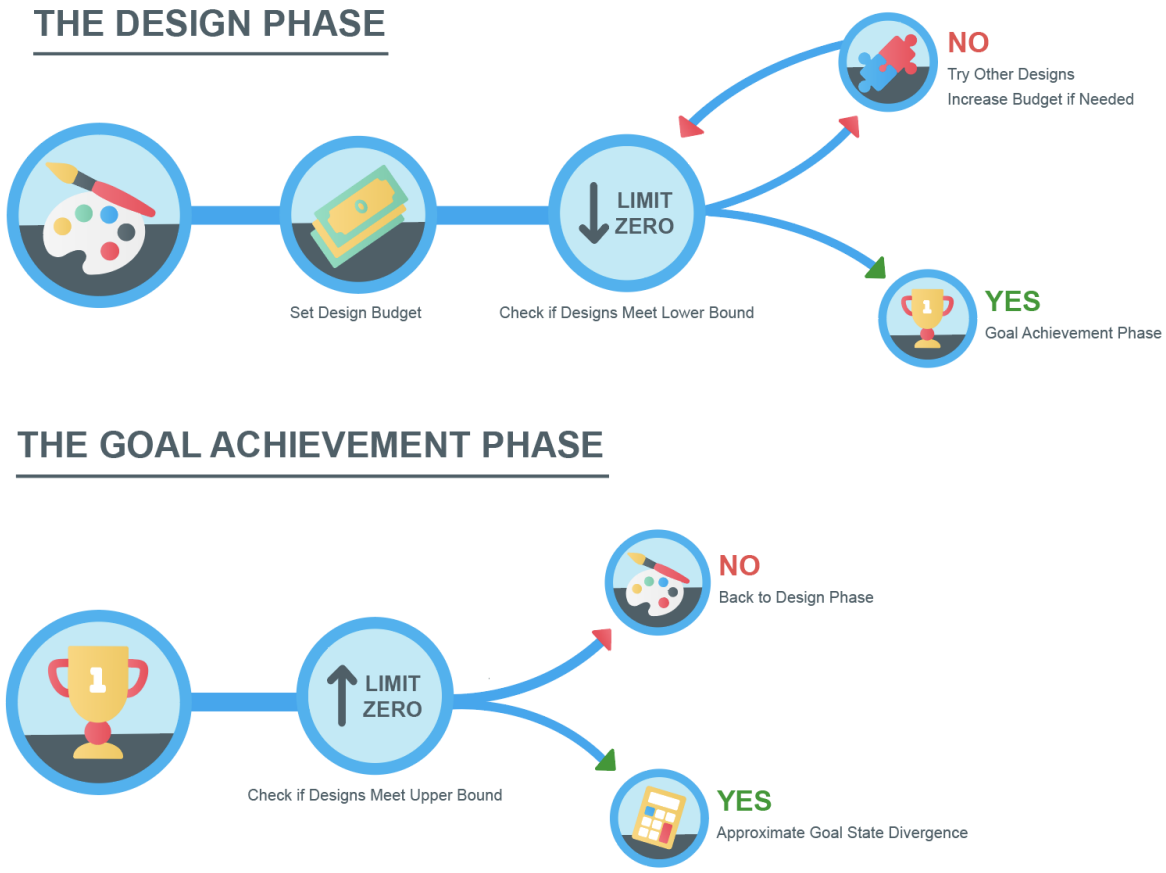


Figure 5.4: In the design phase, a design budget is set and iteratively increased until a set of initial state design modifications are found that will achieve a minimum goal divergence of zero. The updated human/robot models used to accomplish this are then inputted into the goal achievement phase and checked to see if they meet the upper bound. This cycle continues until either both bounds are found and used to approximate Goal State Divergence or no solution is found.

5.4.2 Extended Compiled Model

To find a set of design modifications that fit our budget, and satisfies our (l, u)-bounded requirements, we extend our previously compiled model $\mathcal{M}^\lambda$ into $\mathcal{M}^\lambda_\mathbb{U} = \langle \mathcal{D}^\lambda_\mathbb{U}, \mathcal{I}^\lambda_\mathbb{U}, \mathcal{G}^\lambda_\mathbb{U} \cup \{unseen\_design\}\rangle$. Now, $\mathcal{D}^\lambda_\mathbb{U}$ is defined by the tuple $\mathcal{D}^\lambda_\mathbb{U} = \langle \mathcal{F}^\lambda_\mathbb{U}, \mathcal{A}^\lambda_\mathbb{U}\rangle$, with $\mathcal{F}^\lambda_\mathbb{U}$ containing additional fluents and $\mathcal{A}^\lambda_\mathbb{U}$ more actions. $\mathcal{F}^\lambda_\mathbb{U}$ is now characterized by $\mathcal{F}^\lambda_\mathbb{U} = \mathcal{F}^\lambda \cup \{design\_allowed\} \cup \{unseen\_design\} \cup \mathcal{F}^\tau \cup \mathcal{F}^{\tau+}$, where $\mathcal{F}^\lambda$ corresponds to the same fluents discussed in the previous model compilation section (i.e. $\mathcal{F}^\mathcal{R}$, $\mathcal{F}^\mathcal{H}$, $\mathcal{F}^\theta$ and $\mathcal{F}^\kappa$). $\{design\_allowed\}$ is a fluent used to indicate when design actions can be executed in the design phase, and $\{unseen\_design\}$ is a fluent used to indicate that a given design has not been seen or used before, and may be used to reduce $\mathcal{GSD}$. In the initial state, this fluent begins as true, only flipping to false if a plan tries to use a design seen before. This fluent is important because it guarantees that each design will only be used once. When trying to reduce $\mathcal{GSD}$, $\mathcal{F}^\tau$ and $\mathcal{F}^{\tau+}$ are used to enforce a design budget of size τ. To achieve this, the precondition of all design actions include fluents from $\mathcal{F}^\tau$. Once executed, these fluents are deleted, while fluents from $\mathcal{F}^{\tau+}$ are added. To ensure only τ designs are performed, we update our goal to include $\mathcal{F}^{\tau+}$.

The updated set of actions is shown by $\mathcal{A}^\lambda_\mathbb{U} = ((\mathcal{A}^\lambda \cup \mathcal{A}^\mathbb{U}) \setminus \mathcal{A}^{\kappa-}) \cup \{design\_completed\}$. Like before, $\mathcal{A}^\lambda$ corresponds to the same actions discussed in the previous model compilation section (i.e. $\mathcal{A}^\mathcal{R}$, $\mathcal{A}^\mathcal{H}$, $\mathcal{A}^\theta$ and $\mathcal{A}^\kappa$), while $\mathcal{A}^\mathbb{U} = \tau \times |\mathbb{U}|$ is the available set of design actions that must be executed for a given modification to be made. From this we remove the subset of check disagreement actions, $\mathcal{A}^{\kappa-}$, because for our problem setting, we are only searching for human-robot plans whose final goal states match exactly. Because this only requires check agreement compare actions, we remove any design actions which cause a check disagreement compare action to be executed. When performed, the remaining actions in $(\mathcal{A}^\lambda \cup \mathcal{A}^\mathbb{U})$ result in updates to the human or robot's initial state, per the requirement of each action.

It is important to note that each design action in $\mathcal{A}^{\mathbb{U}}$ corresponds to the addition or removal of an initial state fluent, and a specific time step t_i. As mentioned, these actions can only be performed in the design phase when the $\{design\_allowed\}$ fluent is true. Thus, if an individual design action occurs at time step t_i, $\{design\_allowed,\ f_i\}$ will be its positive precondition, where $f_i \in \mathcal{F}^{\tau}$. Here, only one design action for time step t_i can be performed at a time. Depending on the model update, this action will either add or delete an initial state fluent. Specifically, if a design action makes a fluent true by adding it to the human or robot's initial state, that fluent becomes part of the add effect. In contrast, if a design action removes a fluent from the initial state making it false, that fluent becomes part of the delete effect. This means all design actions will delete $t_i \in \mathcal{F}^{\tau}$, while adding $t_i^{+} \in \mathcal{F}^{\tau+}$ along with the related design. Accompanying this is the $\{design\_completed\}$ action, which ends the design phase by deleting the $\{design\_allowed\}$ fluent after all modifications have been made, and adds the $\{human\_can\_act\}$ fluent so planning can resume. This action also prevents the compilation from returning previously seen designs by using conditional effects to check if they've been examined before. If a design has been seen, these effects delete its $\{unseen\_design\}$ fluent, preventing its use.

For the extended compiled model, the updated initial state is shown by $\mathcal{I}_{\mathbb{U}}^{\lambda} = (\mathcal{I}^{\lambda} \setminus \{human\_can\_act\}) \cup \{unseen\_design, design\_allowed\} \cup \mathcal{F}^{\tau}$. Like before, $\mathcal{I}^{\lambda}$ corresponds to the same initial state discussed in the previous model compilation section (i.e. $\mathcal{I}^{\mathcal{R}}$ and $\mathcal{I}^{\mathcal{H}}$) but now with the $\{human\_can\_act\}$ fluent removed. This is because plans for $\mathcal{M}_{\mathbb{U}}^{\lambda}$ initially start in a design phase, where modifications are being made to reduce $\mathcal{GSD}$, and neither the human nor the robot can act. Here, the $\{design\_allowed\}$ fluent is included to ensure the problem begins in the design phase and the $\{unseen\_design\}$ fluent to guarantee a design has not been seen before in cases where the conditional effects from the *design_completed* action didn't remove it. This changes once the design phase ends and the $\{design\_completed\}$ action is executed, which replaces these fluents with $\{human\_can\_act\}$. The extended

compilation then becomes identical to the previous compilation, and the human and robot can perform actions. To track the design size, $\mathcal{F}^{\tau}$ is also included in the initial state.

Finally, is the updated goal, given by $\mathcal{G}^{\lambda}_{\mathbb{U}} = \mathcal{G}^{\lambda} \cup \mathcal{F}^{\tau+} \cup \{unseen\_design\}$. As before, $\mathcal{G}^{\lambda}$ corresponds to the same set of goals discussed in the previous model compilation section (i.e. $\mathcal{G}^{\mathcal{R}}$, $\mathcal{G}^{\mathcal{H}}$ and $\mathcal{F}^{\kappa}$). However, now $\mathcal{F}^{\tau+}$ has been added to this set, indicating that all τ designs must be applied for the human and robot's goals to fully be achieved. $\{unseen\_design\}$ is also included, indicating that all design modifications used by the planner must not have been seen or used before.

5.4.3 Design Implementation & Cost

When in the design phase, $\mathcal{F}^{\tau+}$ correlates to the number of designs that need to be applied. When implementing these, we keep the cost function the same as before, assigning a unit cost of one to each design modification made. This means when a solution to this problem is found, the set of modifications that return a pair of human-robot plans with zero $\mathcal{GD}^{\downarrow}$ can be automatically found, then checked against the $\mathcal{GD}^{\uparrow}$ requirement. Here, if the requirements are met, we know the cheapest possible set of modifications has been found.

If ever this search fails to return a solution, then we know the design search for the given budget has been exhausted and no other minimal design sets meeting our criteria exist. In such a case, we return to the outer loop to increase our design budget and restart our search for a set of modifications that will reduce $\mathcal{GSD}$ according to our criteria. We detail this process using pseudo-code in Algorithm 1, where the set $found\_designs$ is used to track all previous designs for a given budget that were seen but did not meet the proper criteria, and $all\_designs\_found$ is a flag used to indicate when the design search has been exhausted and the budget needs to be increased. In the appendix of this document, we also provide a more generalized version of this algorithm, with more relaxed assumptions than what's been shown here.

Procedure 1: An algorithm for a HRGAD design problem

Input: $\mathcal{DP}, u$
Output: Model update set $\mathcal{U} \subseteq \mathbb{U}$ that satisfy the requirements that for resulting models $\mathcal{GD}^{\downarrow}$ is zero and $\mathcal{GD}^{\uparrow}$ is u
for τ in 1 ... $|\mathcal{F}^{\mathcal{R}}|$ **do**
  $all\_designs\_found \leftarrow False$
  $found\_designs \leftarrow \{\}$
  while $all\_designs\_found$ is $false$ **do**
    $\mathcal{M}^{\lambda}_{\mathbb{U}} \leftarrow \mathcal{GD}^{\downarrow}\_with\_Design(\mathcal{M}^{\mathcal{R}}, \mathcal{M}^{\mathcal{H}}, \tau, found\_designs)$
    $\pi^{\lambda}_{\mathbb{U}} \leftarrow GetPlan(\mathcal{M}^{\lambda}_{\mathbb{U}})$
    if $\pi^{\lambda}_{\mathbb{U}}$ length is 0 **then**
      $all\_designs\_found \leftarrow True$
    else
      Extract model updates $\mathcal{U}$ from $\pi^{\lambda}_{\mathbb{U}}$ and add it to $found\_designs$
      if $\mathcal{GD}^{\uparrow}(\Lambda(\mathcal{M}^{\mathcal{R}},\mathcal{U}), \Lambda(\mathcal{M}^{\mathcal{H}},\mathcal{U}))$ is u **then**
        return $\mathcal{U}$
      end if
    end if
  end while
end for

Chapter 6

EVALUATION & RESULTS

We now direct our attention to how we empirically evaluated these ideas. For our evaluation, we implemented and computationally characterized six compilations in total. This included three baseline approaches including the $\mathcal{GD}^{\uparrow}$, $\mathcal{GD}^{\downarrow}$, and $\mathcal{GD}^{\downarrow}$ with design, and three primary approaches involving design, consisting of `Main`, `Main-fl`, and `Naïve`. In this section, we introduce the datasets used to accomplish this, along with more details related to these compilations and our methodology.

Domain	IPC Year	Description
Blocksworld	2000	A domain where a robot arm is tasked with re-assembling a series of blocks on a table. Here, the arm can pick up, put down, stack, or unstack the blocks. To achieve its goal, the robot arm must successfully stack the blocks in a certain way.
Depot	2002	This domain combines the Blocksworld and Logistics domains together. Here, trucks transport crates to a desired location, where they are then hoisted onto pallets. To successfully do this, all crates must be delivered to the proper place and stacked on the correct pallet.
Elevator	2002	In a building with multiple stories and elevators, passengers must be transported to their destination floor. To successfully do this, all passengers must end up on their desired floor, with the system deciding when they can and cannot exit the elevator.
Logistics	2000	A domain where packages are delivered in cities using trucks and between cities using airplanes. Using these methods, all packages must be delivered to the correct location.
Zenotravel	2002	A transportation domain where people are moved around in planes going fast and slow. When fast movement is used, more fuel is consumed then with slow movement. The optimal plan involves transporting people to their desired destination without using too much fuel.

Table 6.1: Here we describe the domains each set of problem instance were taken from for use in our experiments

6.1 International Planning Competition (IPC) Datasets

Our dataset consisted of instances taken from five standard International Planning Competition (IPC) domains[1,2]: Blocksworld, Logistics, Zenotravel, Elevator and Depot. In Table 6.1, we describe each of these domains in more detail. From each domain, five problem instances were randomly selected based on the size of their initial state. Here, instances with smaller initial states were chosen over those with larger ones, to help minimize planner issues and problem run-time.

To convert these five instances into a goal divergence problem, a duplicate of each problem instance was made. All original problem instances were kept the same and used to represent the robot's model. From all duplicate problem instances, five random initial state fluents were deleted. These were used to represent the human's model. This ensured that the robot's actual state and capabilities differed from the human's beliefs about them, while also guaranteeing that five design modifications could always be made to lower the $\mathcal{GSD}$ between them. From the Zenotravel domain, the zoom action was removed to avoid large differences between its fuel-level fluents, while all other domains were kept the same. To ensure all plans ended in the same goal state, the goal specification of all problem instances was updated to match. This guaranteed that all problems were within the required $\mathcal{GD}^{\uparrow}$ limit before any experiments were run. In all, five problem variations for each domain instance were created (i.e. 25 instances per domain, including the problem and domain files), for a total of 125 instances created across all domains.

[1]https://github.com/potassco/pddl-instances/tree/master/ipc-2000

[2]https://github.com/potassco/pddl-instances/tree/master/ipc-2002

6.2 Compilation Objectives

For all compilations, our objective was to maintain consistency between bounds while attaining zero divergence between the robot and human's models. To achieve this, we only considered $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$ with limits of zero. For the $\mathcal{GD}^{\uparrow}$, this was particularly significant because it allowed us to check if the compilation was unsolvable should one or more check disagreement actions exist between the robot and human's models. Further, this consistency allowed us to perform cross-domain comparisons, while avoiding the use of more costly, cost-optimal planners.

6.3 Main & Baseline Compilations

For our primary approaches, three different compilations were run. The first was our `Main` compilation, which used enforced ordering between the human and robot's actions and was the same as the algorithm described in Section 5.4. Next was a nearly identical variation of this algorithm, called `Main-fl`. This flattened compilation executed a near identical sequence of steps to `Main`, however, it did not enforce an ordering between the human and robot's actions. This means all problems run using this algorithm, began with both the $\{human\_can\_act\}$ and $\{robot\_can\_act\}$ fluents as part of their initial state. Last was our `Naïve`, breadth-first search baseline algorithm, which did not leverage planning to identify design modifications. Instead, it iterated over all possible design modifications and tested them to see which might result in upper and lower bounds of zero.

To reveal how efficient different parts of our compilation were running, we also individually computed the $\mathcal{GD}^{\uparrow}$, $\mathcal{GD}^{\downarrow}$, and $\mathcal{GD}^{\downarrow}$ with design. Here, the $\mathcal{GD}^{\uparrow}$ was used to compute the upper bound or worst-case divergence. $\mathcal{GD}^{\downarrow}$ was a baseline approach which implemented a simple breadth-first search to identify designs to find a lower bound or best-case divergence of zero. And the $\mathcal{GD}^{\downarrow}$ with design was an updated compilation which did the same thing, but

used planning to identify these designs. Here, the $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$ with design corresponded to the compilation pieces used in the primary `Main` and `Main-fl` methods. While the $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$ corresponded to those used in the `Naïve` baseline.

6.4 Setup & Experimental Approach

For our evaluation, we found the average and standard deviation time in seconds taken for each compilation to be solved by averaging across all human-robot model pairs. We focused on this metric because it had the biggest differential across all our baselines and was relevant given the nature of some of our baselines (i.e. `Naïve`). Additionally, for other researchers wanting to test our methods on their domain of interest, runtime is a useful metric. We show these results in Table 6.2, where all numbers have been rounded to the nearest hundredth.

It is important to note that for our primary compilations (i.e. `Main`, `Main-fl`, and `Naïve`), this means the times listed reflect how long it took to find a minimal set of design modifications that would ensure a $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$ of zero. This required that these bounds be solved for multiple times, resulting in longer runtimes. For the baseline approaches (i.e. $\mathcal{GD}^{\uparrow}$, $\mathcal{GD}^{\downarrow}$, and $\mathcal{GD}^{\downarrow}$ with design), this meant solving for only one bound with the human-robot ordering constraints enforced.

To find a suitable plan for each compilation, the Lama planner was used to run our experiments [48]. We did this using a Mac computer with an Apple M2 Max chip and 64 GB of RAM. Each experiment was allowed to run for 60 minutes at which point either a solution was found, or the test was purposely ended.

6.5 Results

In analyzing our results (Table 6.2), we found that among our primary methods, the `Main` and `Main-fl` algorithms took the shortest amount of time to run, especially when compared with our baseline methods. Here, the captured times were nearly identical, with only a few small variations between instances reported for each. In some cases, enforcing an ordering between the robot and human's actions, did provide a slight improvement in time, compared to the flattened compilation. This is most likely due to the reduced branching factor provided by this ordering. In comparing these times to the `Naïve` approach, we find the `Main` and `Main-fl` methods were significantly faster, especially when considering the Depot domain. Presumably, this is due to both methods leveraging planning to identify possible design modifications, resulting in faster runtimes. Among all compilations, we also see that the Elevator domain took the least amount of time to run.

In looking at our baseline methods, we find that the $\mathcal{GD}^{\uparrow}$ in general had higher runtimes then the $\mathcal{GD}^{\downarrow}$. This is unsurprising, however, considering that in this scenario, the $\mathcal{GD}^{\uparrow}$ was being tested for unsolvability. In comparing the $\mathcal{GD}^{\downarrow}$ to $\mathcal{GD}^{\downarrow}$ with design, we see that adding design into the compilation added a slight overhead.

Domain	`Main`	`Main-fl`	`Naive`	$\mathcal{GD}^{\downarrow}$	$\mathcal{GD}^{\downarrow}$ `with Design`	$\mathcal{GD}^{\uparrow}$
Blocksworld	73.849 ± 2.150	73.172 ± 2.281	387.178 ± 6.724	11.703 ± 1.264	12.955 ± 0.469	12.642 ± 1.461
	71.966 ± 3.183	74.115 ± 3.570	386.627 ± 4.365	11.674 ± 1.165	11.739 ± 2.044	12.893 ± 2.097
	111.919 ± 30.519	110.049 ± 27.237	432.541 ± 24.484	12.420 ± 5.547	11.883 ± 0.250	30.181 ± 9.809
	97.049 ± 1.961	96.051 ± 1.213	417.206 ± 3.636	11.942 ± 0.760	12.748 ± 0.595	35.035 ± 1.308
	118.487 ± 28.660	116.327 ± 30.162	453.887 ± 21.167	13.038 ± 6.836	12.505 ± 0.529	25.932 ± 12.534
Depot	35.593 ± 2.338	31.877 ± 2.929	188.556 ± 10.304	5.747 ± 1.517	5.234 ± 1.096	5.006 ± 0.465
	86.355 ± 0.810	85.794 ± 1.029	448.649 ± 4.008	13.530 ± 0.761	13.991 ± 0.293	16.285 ± 0.564
	84.901 ± 0.976	84.861 ± 1.396	446.248 ± 0.784	13.463 ± 0.633	14.149 ± 0.860	15.288 ± 0.604
	85.238 ± 1.170	85.853 ± 0.528	445.714 ± 2.721	13.455 ± 0.702	13.601 ± 0.368	15.479 ± 0.673
	86.531 ± 2.873	84.731 ± 1.181	452.672 ± 3.351	13.648 ± 0.668	14.428 ± 1.613	15.723 ± 0.205
Elevator	3.691 ± 0.013	3.629 ± 0.009	17.930 ± 0.052	0.543 ± 0.008	0.607 ± 0.008	0.561 ± 0.001
	4.116 ± 0.013	4.070 ± 0.008	19.708 ± 0.018	0.597 ± 0.011	0.676 ± 0.011	0.610 ± 0.001
	4.119 ± 0.009	4.085 ± 0.020	19.771 ± 0.072	0.599 ± 0.011	0.674 ± 0.002	0.611 ± 0.002
	4.123 ± 0.011	4.066 ± 0.013	19.843 ± 0.065	0.601 ± 0.011	0.673 ± 0.002	0.615 ± 0.003
	4.118 ± 0.008	4.068 ± 0.006	19.796 ± 0.063	0.600 ± 0.010	0.672 ± 0.002	0.611 ± 0.001
Logistics	33.062 ± 0.738	32.330 ± 0.337	50.306 ± 0.582	0.886 ± 2.536	0.807 ± 0.017	28.785 ± 0.755
	31.936 ± 0.495	30.381 ± 0.158	48.112 ± 0.643	0.684 ± 0.026	0.814 ± 0.016	27.577 ± 0.485
	30.457 ± 0.408	30.457 ± 0.209	47.927 ± 0.554	0.676 ± 0.023	0.804 ± 0.009	26.194 ± 0.393
	32.795 ± 0.488	32.824 ± 0.552	50.068 ± 0.469	0.684 ± 0.024	0.819 ± 0.018	28.455 ± 0.467
	30.439 ± 0.521	30.641 ± 0.414	48.040 ± 0.403	0.683 ± 0.023	0.806 ± 0.015	26.152 ± 0.469
Zenotravel	5.084 ± 0.025	5.025 ± 0.017	23.641 ± 0.070	0.716 ± 0.009	0.791 ± 0.005	0.745 ± 0.005
	5.167 ± 0.024	5.142 ± 0.019	24.022 ± 0.157	0.728 ± 0.013	0.807 ± 0.002	0.755 ± 0.003
	7.025 ± 0.041	6.953 ± 0.045	31.263 ± 0.160	0.944 ± 0.016	1.044 ± 0.012	1.081 ± 0.004
	7.210 ± 0.066	7.164 ± 0.066	31.468 ± 0.126	0.949 ± 0.017	1.063 ± 0.009	1.148 ± 0.010
	10.021 ± 0.662	9.986 ± 0.636	40.853 ± 8.818	1.367 ± 0.026	1.496 ± 0.010	1.510 ± 0.015

Table 6.2: The average and standard deviation time taken by each method compared to each baseline in seconds per instance. The first three columns respectively present the time taken by our method, a variation of our method that doesn't enforce ordering, and a baseline that iterates over possible designs. The final three columns report the average time taken to compute the lower bound of $\mathcal{GSD}$, lower bound with design, and upper bound.

Chapter 7
DISCUSSION & FUTURE WORK

To the best of our knowledge, we are the first to use design to better align a robot's final goal state with a human's expectations. Because of this, we are unaware of any pre-existing baselines or data points that we could have used to measure this work against. For this reason, in our evaluation, we instead compared results from our primary methods against baselines we computed on our own. However, as this work evolves in the future, making these comparisons as the data is available may provide additional insights into this work.

During our experiments, it's important to note that we only allowed initial state design modifications to be made to the robot and human's models. This allowed us to provide a more compact and concise compilation, that relied on the least number of assumptions. Additionally, given our specific problem, this was necessary because apart from the initial state, all other parts of the human and robot's models were identical. In the future, testing our methods on more complex decision-making frameworks where differences between the robot and human's models extend beyond their initial states and a wider variety of design modifications exist for use, would be a valuable extension of this work.

Recall, in our experiments, all design modifications were assigned a unit cost. Future work might also consider assigning different costs to different design modifications, to see what effect this would have on the human and robot's plan selections and final goal states. Additionally, within this study, we assume all design modifications can be made independent of each other. Subsequent investigations into $\mathcal{GSD}$ could challenge this assumption by enforcing a dependency between some or all design modifications that could be applied.

Similarly, in this work all fluents were treated the same but in a real-world setting, some fluents may contribute more heavily to a robot ending in an undesirable, potentially dangerous final goal state then others. To avoid this, exploring more methods that could

better differentiate between the fluents which may cause benign versus hazardous side effects could add an additional, safety-focused dimension to this work.

Lastly, many alternate approaches to $\mathcal{GSD}$ exist for exploration, outside of what we presented here. For example, experiments could be run on scenarios where a robot is assigned multiple goals at one time or given more complex objectives with specific preferences. For a more in-depth analysis, various forms of temporal logic or reward functions could be included as part of these experiments. Future $\mathcal{GSD}$ experiments could also be run in environments where numerous humans, robots, or some combination of the two are simultaneously operating. Looking at how these problems may be solved in isolation or in more dynamic environments outside of the static ones we consider here, might also elevate this research.

Chapter 8
CONCLUSION

For human-robot collaboration to be successful, a robot must reach a final goal state aligned with a human's expectations. When this fails to happen, at best a human's goal may not be achieved, and at worst the robot may end in a final, negative side-effect producing state. To solve this problem, in this work, I presented the first-ever design framework for generating behavior congruent with a human's goal-state expectations, using environment design. Though many problems in this space exist, I specifically explored how to quantify the divergence between a pair of human-robot goal states using my novel Goal State Divergence metric and reduce it using the smallest set of environment modifications possible. While wider experimentation is needed to truly understand the efficacy of my results, my methods were able to achieve perfect goal state alignment between the human and robot for all domains tested, highlighting not only the potential of this work but laying a solid foundation for other researchers in the AI community to build upon.

Bibliography

[1] M. Mechergui and S. Sreedharan, "Goal alignment: Re-analyzing value alignment problems using human-aware ai," in *Proceedings of the 2023 international conference on autonomous agents and multiagent systems*, 2023, pp. 2331–2333.

[2] Y. Zhang, S. Sreedharan, A. Kulkarni, T. Chakraborti, H. H. Zhuo, and S. Kambhampati, "Plan explicability and predictability for robot task planning," in *2017 ieee international conference on robotics and automation (icra)*, 2017, pp. 1313–1320.

[3] T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati, "Plan explanations as model reconciliation: Moving beyond explanation as soliloquy," in *International joint conference on artificial intelligence*, 2017, pp. 156–163.

[4] H. Zhang and D. Parkes, "Value-based policy teaching with active indirect elicitation," in *Proceedings of the 23rd national conference on artificial intelligence*, vol. 1, 2008, pp. 208–214.

[5] S. Keren, A. Gal, and E. Karpas, "Goal recognition design in deterministic environments," *Journal of artificial intelligence research*, vol. 65, no. 1, pp. 209–269, 2019.

[6] H. Zhang, Y. Chen, and D. Parkes, "A general approach to environment design with one agent," in *Proceedings of the 21st international joint conference on artificial intelligence*, 2009, pp. 2002–2008.

[7] A. M. Metelli, *Exploiting environment configurability in reinforcement learning*. IOS Press, 2022.

[8] G. Yu and C.-J. Ho, "Environment design for biased decision makers," in *Proceedings of the thirty-first international joint conference on artificial intelligence*, 2022, pp. 592–598.

[9] S. Keren, A. Gal, and E. Karpas, "Goal recognition design," in *Proceedings of the twenty-fourth international conferenc on international conference on automated planning and scheduling*, 2014, pp. 154–162.

[10] R. Mirsky, K. Gal, R. Stern, and M. Kalech, "Goal and plan recognition design for plan libraries," *Acm transactions on intelligent systems and technology*, vol. 10, no. 2, 2019.

[11] S. Keren, A. Gal, and E. Karpas, "Goal recognition design - survey," in *Proceedings of the twenty-ninth international joint conference on artificial intelligence*, 2020, pp. 4847–4853.

[12] S. Keren, A. Gal, E. Karpas, L. Pineda, and S. Zilberstein, "Redesigning stochastic environments for maximized utility," in *Proceedings of the aaai conference on artificial intelligence*, vol. 31, 2017.

[13] S. Keren, L. Pineda, A. Gal, E. Karpas, and S. Zilberstein, "Equi-reward utility maximizing design in stochastic environments," 2017, pp. 4353–4360.

[14] S. Keren, L. Pineda, A. Gal, E. Karpas, and S. Zilberstein, "Efficient heuristic search for optimal environment redesign," in *Proceedings of the international conference on automated planning and scheduling*, vol. 29, 2021, pp. 246–254.

[15] A. M. MacNally, N. Lipovetzky, M. Ramirez, and A. R. Pearce, "Action selection for transparent planning," in *Proceedings of the 17th international conference on autonomous agents and multiagent systems*, 2018, pp. 1327–1335.

[16] M. Kwon, S. H. Huang, and A. D. Dragan, "Expressing robot incapability," in *Proceedings of the 2018 acm/ieee international conference on human-robot interaction*, 2018, pp. 87–95.

[17] T. Chakraborti, S. Sreedharan, S. Grover, and S. Kambhampati, "Plan explanations as model reconciliation: An empirical study," in *Proceedings of the 14th acm/ieee international conference on human-robot interaction*, 2020, pp. 258–266.

[18] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial intelligence*, vol. 267, pp. 1–38, 2019.

[19] S. Kraus, A. Azaria, J. Fiosina, *et al.*, "Ai for explaining decisions in multi-agent environments," vol. 34, pp. 13 534–13 538, 2020.

[20] S. Sreedharan, S. Srivastava, and S. Kambhampati, "Hierarchical expertise level modeling for user specific contrastive explanations," in *Proceedings of the 27th international joint conference on artificial intelligence*, 2018, pp. 4829–4836.

[21] B. Seegebarth, F. Müller, B. Schattenberg, and S. Biundo, "Making hybrid plans more clear to human users - a formal approach for generating sound explanations," in *Proceedings of the international conference on automated planning and scheduling*, vol. 22, 2012, pp. 225–233.

[22] S. Sreedharan, T. Chakraborti, and S. Kambhampati, "Handling model uncertainty and multiplicity in explanations via model reconciliation," in *Proceedings of the international conference on automated planning and scheduling*, vol. 28, 2018, pp. 518–526.

[23] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *Corr*, 2016. arXiv: 1606.06565.

[24] D. Weld and O. Etzioni, "The first law of robotics (a call to arms)," in *Proceedings of the twelfth aaai national conference on artificial intelligence*, 1994, pp. 1042–1047.

[25] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg, "AI safety gridworlds," *Corr*, 2017. arXiv: 1711.09883.

[26] S. Saisubramanian, S. Zilberstein, and E. Kamar, "Avoiding negative side effects due to incomplete knowledge of ai systems," *Ai magazine*, vol. 42, no. 4, pp. 62–71, 2021.

[27] D. Hadfield-Menell, A. Dragan, P. Abbeel, and S. Russell, "Cooperative inverse reinforcement learning," in *Proceedings of the 30th international conference on neural information processing systems*, 2016, pp. 3916–3924.

[28] R. Shah, D. Krasheninnikov, J. Alexander, P. Abbeel, and A. Dragan, "Preferences implicit in the state of the world," in *Proceedings of the 7th international conference on learning representations*, 2019.

[29] C. Basich, J. Svegliato, K. H. Wray, S. Witwicki, J. Biswas, and S. Zilberstein, "Learning to optimize autonomy in competence-aware systems," in *International foundation for autonomous agents and multiagent systems*, 2020, pp. 123–131.

[30] S. Zhang, E. Durfee, and S. Singh, "Querying to find a safe policy under uncertain safety constraints in markov decision processes," in *Proceedings of the aaai conference on artificial intelligence*, vol. 34, 2020, pp. 2552–2559.

[31] V. Krakovna, L. Orseau, R. Kumar, M. Martic, and S. Legg, *Penalizing side effects using stepwise relative reachability*, 2018.

[32] S. Zhang, E. H. Durfee, and S. Singh, "Minimax-regret querying on side effects for safe optimality in factored markov decision processes," in *Proceedings of the 27th international joint conference on artificial intelligence*, 2018, pp. 4867–4873.

[33] S. Saisubramanian and S. Zilberstein, "Mitigating negative side effects via environment shaping," in *Proceedings of the 20th international conference on autonomous agents and multiagent systems*, 2021, pp. 1640–1642.

[34] A. Kulkarni, Y. Zha, T. Chakraborti, S. G. Vadlamudi, Y. Zhang, and S. Kambhampati, "Explicable planning as minimizing distance from expected behavior," in *Proceedings of the 18th international conference on autonomous agents and multiagent systems*, 2019, pp. 2075–2077.

[35] T. Chakraborti, S. Sreedharan, and S. Kambhampati, "Balancing explicability and explanations in human-aware planning," in *Proceedings of the twenty-eighth international joint conference on artificial intelligence*, 2019, pp. 1335–1343.

[36] S. Sreedharan, T. Chakraborti, C. Muise, and S. Kambhampati, "Expectation-Aware Planning: A Unifying Framework for Synthesizing and Executing Self-Explaining Plans for Human-Aware Planning," in *Proceedings of the aaai conference on artificial intelligence*, vol. 34, 2020, pp. 2518–2526.

[37] J. F. Fisac, C. Liu, J. B. Hamrick, S. S. Sastry, J. K. Hedrick, T. L. Griffiths, and A. D. Dragan, "Generating plans that predict themselves," in *Workshop on the algorithmic foundations of robotics*, 2018.

[38] A. Hanni, A. Boateng, and Y. Zhang, *Safe explicable robot planning*, 2023. arXiv: 2304.03773.

[39] A. Kulkarni, S. Sreedharan, S. Keren, T. Chakraborti, D. E. Smith, and S. Kambhampati, "Design for interpretability," in *Icaps workshop on explainable ai planning*, 2019.

[40] A. Kulkarni, S. Sreedharan, S. Keren, T. Chakraborti, D. E. Smith, and S. Kambhampati, "Designing environments conducive to interpretable robot behavior," in *2020 ieee/rsj international conference on intelligent robots and systems (iros)*, 2020, pp. 10 982–10 989.

[41] H. Geffner and B. Bonet, *A concise introduction to models and methods for automated planning: synthesis lectures on artificial intelligence and machine learning*, 1st ed. Morgan & Claypool Publishers, 2013.

[42] M. Ghallab, A. Howe, C. Knoblock, D. Mcdermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, *PDDL—The Planning Domain Definition Language*, 1998.

[43] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*, 3rd ed. Prentice Hall, 2010.

[44] E. Callanan, R. D. Venezia, V. Armstrong, A. Paredes, T. Chakraborti, and C. Muise, *Macq: A holistic view of model acquisition techniques*, 2022. eprint: 2206.06530.

[45] S. Sreedharan, A. O. Hernandez, A. P. Mishra, and S. Kambhampati, "Model-free model reconciliation," in *Proceedings of the 28th international joint conference on artificial intelligence*, 2019, pp. 587–594.

[46] D. Hutto and I. Ravenscroft, "Folk Psychology as a Theory," in *The Stanford encyclopedia of philosophy*, E. N. Zalta, Ed., Fall 2021, Metaphysics Research Lab, Stanford University, 2021.

[47] U. Soni, S. Sreedharan, and S. Kambhampati, "Not all users are the same: Providing personalized explanations for sequential decision making problems," in *2021 ieee/rsj international conference on intelligent robots and systems (iros)*, 2021, pp. 6240–6247.

[48] S. Richter and M. Westphal, "The LAMA planner: Guiding cost-based anytime planning with landmarks," *Journal of artificial intelligence research*, vol. 39, pp. 127–177, 2010.

[49] M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel, "Coming up with good excuses: What to do when no plan can be found," in *Proceedings of the international conference on automated planning and scheduling*, vol. 20, 2010, pp. 81–88.

Appendix A
UPDATED COMPILATION

In this section, we discuss how the methods presented in this work can be generalized to limit the min and max to a set of plans and share an updated compilation to support design changes made to any part of the model.

A.0.1 Theoretical Properties

Proposition 5. *Algorithm 1 is a sound procedure for finding a design ξ that will result in a model where $\mathcal{GD}^{\downarrow}(\mathcal{M}_{\xi}^{\mathcal{R}}, \mathcal{M}_{\xi}^{\mathcal{H}}) = 0$ and $\mathcal{GD}^{\uparrow}(\mathcal{M}_{\xi}^{\mathcal{R}}, \mathcal{M}_{\xi}^{\mathcal{H}}) \leq u$, where $\mathcal{M}_{\xi}^{\mathcal{R}} = \Lambda(\mathcal{M}^{\mathcal{R}}, \xi)$ and $\mathcal{M}_{\xi}^{\mathcal{H}} = \Lambda(\mathcal{M}^{\mathcal{H}}, \xi)$*

.

The proof of the above proposition directly follows from the soundness of the individual compilation leveraged in the algorithm. The soundness of the basic compilations is established in Propositions 1 and 2 in the main paper. The only additional extension considered is the fact that the $\mathcal{GD}^{\downarrow}$ compilation considered is one that also identifies the design. Here, the compilation corresponds to two components: the preliminary set of actions that identifies a set of designs and the part that checks the resultant problems $\mathcal{GD}^{\downarrow}$. Now, the former component is similar to previous works that have looked at planning compilations that make problems solvable (cf. [49]). In this case, the second part will only solve a solvable problem if $\mathcal{GD}^{\downarrow}$ is zero. As such, the overall algorithm will only identify a valid design.

Proposition 6. *Algorithm 1 is a complete and optimal procedure, so far that it will always find a valid design ξ that meets the criteria for the minimal solution provided by Definition 6 for $\ell = 0$ and u, provided one exists.*

Note that the previous proposition establishes that the procedures identify sound designs. Now the compilation ensures that the $\mathcal{GD}^{\downarrow}$ procedure never repeats the same design. Since the compilation only allows for a fixed design size, the number of iterations here would be finite. Provided the planner we use is complete, it will iterate over all possible designs that can result in $\ell = 0$. Since the possible τ from one to $|\mathcal{F}^{\mathcal{R}}|$, it will always find the smallest possible design. It's worth noting that there can never be a minimal design larger than $|\mathcal{F}^{\mathcal{R}}|$ since the latter corresponds to a design where all possible values of all fluents are changed. As such, the procedure is guaranteed to always return the minimal solution if one exists.

A.1 Exposition on the Compilations

A.1.1 Supporting Plan Subsets

As discussed, the definitions provided in the main paper are quite relaxed in the plans being considered. By considering the space of all plans we are effectively considering a much larger space of models than what would ever be expected by a human or executed by the robot. This will result in weaker bounds than what might be effective. For the robot plan, one of the most direct considerations we can make is to effectively restrict the space of plans to just the optimal plans. However, to make assertions about the possible human plans, we will need to associate a decision-making model with the human. A popular option that is widely used in human-AI interaction literature is the noisy-rational model, where the likelihood of the human choosing a plan is given as

$$P(\pi) \propto \epsilon^{-\beta \times c(\pi)}$$

Where β is a parameter called the rationality parameter. Now we can only consider plans that are above some probability threshold ϵ. For any such probability threshold, there will

be a corresponding cost value C_ϵ, such that for any plan π, where $c(\pi) > C_\epsilon$, you will have $P(\pi) < \epsilon$. We will denote the corresponding upper and lower bounds of $\mathcal{GSD}$, where the plan spaces are bounded as described above as $\mathcal{GD}^{\uparrow}_{\langle\epsilon,*\rangle}$ and $\mathcal{GD}^{\downarrow}_{\langle\epsilon,*\rangle}$.

We can form variations of Definition 6 provided in the main paper, by replacing $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$, with $\mathcal{GD}^{\uparrow}_{\langle\epsilon,*\rangle}$ and $\mathcal{GD}^{\downarrow}_{\langle\epsilon,*\rangle}$.

A.1.1.1 Calculating these new bounds

To calculate $\mathcal{GD}^{\uparrow}_{\langle\epsilon,*\rangle}$ and $\mathcal{GD}^{\downarrow}_{\langle\epsilon,*\rangle}$, we can place additional constraints on the human part of the plan to restrict plans of cost higher than the cost C_ϵ. While in the most general case, this would involve additional book-keeping, one of the special cases, namely restricting to just optimal plans in the human model (represented as $\mathcal{GD}^{\uparrow}_{*}$ and $\mathcal{GD}^{\downarrow}_{*}$) allows for a surprisingly simple formulation.

Proposition 7. *For a given compiled model $\mathcal{M}^\lambda$, let us adopt the following cost function*

- *Cost of the human and robot action copies follows the ordering in the original models. For human action copies, this means, for any two actions $a_1^{\mathcal{H}'}, a_2^{\mathcal{H}'} \in \mathcal{A}^{\mathcal{H}'}$, $C^\lambda(a_1^{\mathcal{H}'}) < C^\lambda(a_2^{\mathcal{H}'})$, if and only if, $C^{\mathcal{H}}(a_1^{\mathcal{H}}) < C^{\mathcal{H}}(a_2^{\mathcal{H}})$, where $a_1^{\mathcal{H}}, a_2^{\mathcal{H}} \in \mathcal{A}^{\mathcal{H}}$ are the corresponding human actions. The same ordering constraint holds for $\mathcal{A}^{\mathcal{R}'}$ and $\mathcal{A}^{\mathcal{R}}$*
- *Cost of the cheapest human action is higher than the costs of the costliest robot plan, i.e., $min_{a\in\mathcal{A}^{\mathcal{H}'}}(C^\lambda(a)) > max_{a\in\mathcal{A}^{\mathcal{R}'}}(C^\lambda(a)) \times 2^{|\mathcal{F}^{\mathcal{R}}|}$*
- *Cost of the cheapest robot action is higher than the sum of the costs of all check agreement actions, i.e. $min_{a\in\mathcal{A}^{\mathcal{R}'}}(C^\lambda(a)) > \sum_{f_i^{\mathcal{R}}\in\mathcal{F}^{\mathcal{R}}} \mathcal{P}_1$*
- *Cost of each check agreement action is unit cost (i.e. $\mathcal{P}_2 = 1$) and disagreement is $\mathcal{P}_2 = 0$.*

For the given cost function, let π^λ be an optimal plan, then $\mathcal{GD}^{\uparrow}_{*}(\mathcal{M}^{\mathcal{R}}, \mathcal{M}^{\mathcal{H}}) = |\kappa^{-}(\pi^\lambda)|$.

Proof Sketch. The cost function employed ensures that no optimal plan can contain a suboptimal human plan in it. Among all the solutions where $\mathcal{H}(\pi^\lambda)$ corresponds to an optimal plan, the cost is dominated by robot actions. Thus no amount of agreement or disagreement would justify the selection of a suboptimal plan for $\mathcal{R}(\pi^\lambda)$. Finally, among the optimal robot and human plans, it selects ones that minimize the use of check agreement actions. □

One can also calculate $\mathcal{GD}^{\downarrow}_{*}$, by using a cost function that is pretty much the same as the one described above, except now $\mathcal{P}_1 = 0$ and $\mathcal{P}_2 > 2^{|\mathcal{F}^{\mathcal{R}}|}$.

A.1.2 Updated Compilation

We had already provided a compilation in the paper that supports the automatic identification of designs for conditions where $\ell = 0$ and the design corresponds to merely initial state updates. Now we will provide a more general formulation that relaxes these two assumptions.

As with the previous formulation, each design now corresponds to a specific model component. Though now the component can be any part of the model other than the goals. Changes to the initial state are treated as before by actions that directly update the initial state before the *design_completed* action. For the other model changes, we will again introduce a set of design actions $\mathcal{A}^{\mathcal{D}'}$, that will possibly add a fluent each from the set $\mathcal{D}'_{\mathbb{U}}$. As with previous design actions, we constrain the applicability of these actions to only before the *design_completed* action. Now for each model component that will be influenced by these designs, we will introduce a new model component in the model.

These new model components will be conditioned on the new design fluents. For example, let's say a precondition p is added to action a by a design corresponding to fluent $d_i \in \mathcal{A}^{\mathcal{D}'}$, then the human and robot copies of the action $a^{\mathcal{H}}$ and $a^{\mathcal{R}}$ will now contain a precondition $\neg d_i \vee p^{\mathcal{H}}$ and $\neg d_i \vee p^{\mathcal{R}}$ respectively. If the design results in a precondition being removed, the precondition becomes $d_i \vee p^{\mathcal{H}}$ in the human model (similar precondition will be introduced in the robot model). Similarly, for effects, we can introduce conditional effects, where it is

conditioned on design fluents from $\mathcal{D}'_{\mathbb{U}}$. Now if we keep the rest of the compilation the same, we can support designs that update any part of the model.

The next generalization we talked about was to support a non-zero ℓ. To do this, we will introduce a new set of fluents $\mathcal{F}^{\ell}$, where $|\mathcal{F}^{\ell}| = \ell$, is used to track the budget. To ensure that we can only allow ℓ possible disagreements, we re-introduce the disagreement actions. Now we will have $|\mathcal{F}| \times \ell$ copies. One for each fluent and will use up one of the possible allotted budgets. To capture this, we will add to the precondition the i^{th} copy of any disagreement action for fluent f, the fluent f_i^{ℓ} (from the set $\mathcal{F}^{\ell}$). The disagreement action will also delete this fluent. So at most ℓ disagreement actions can be used in any given plan.

A.2 Additional Experiments

We were particularly interested in two additional sets of analyses of the algorithm. The first measures how the properties change with respect to the size of the problem, and the other one is how the design size affects the best $\mathcal{GSD}$ found.

A.2.1 Effect of Problem Size on the Time Taken

First, we will start by looking at the sizes of the problems considered in our evaluation. In particular, we will focus on the number of grounded predicates since the planner we consider operates over grounded planning problems and since the count gives a direct size of the state space. This number was measured after a reachability analysis. Table A.1 presents these numbers. Figures A.1, A.2, and A.3 provide an overview of the time taken as plotted against the size of the problem. In addition to the total average time taken versus the time taken for individual $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$ calculations. We see the effect of the size more clearly in the latter calculation time. We see that apart from logistics, there is some variation in the number of grounded predicates, with the most variation occurring in Zenotravel.

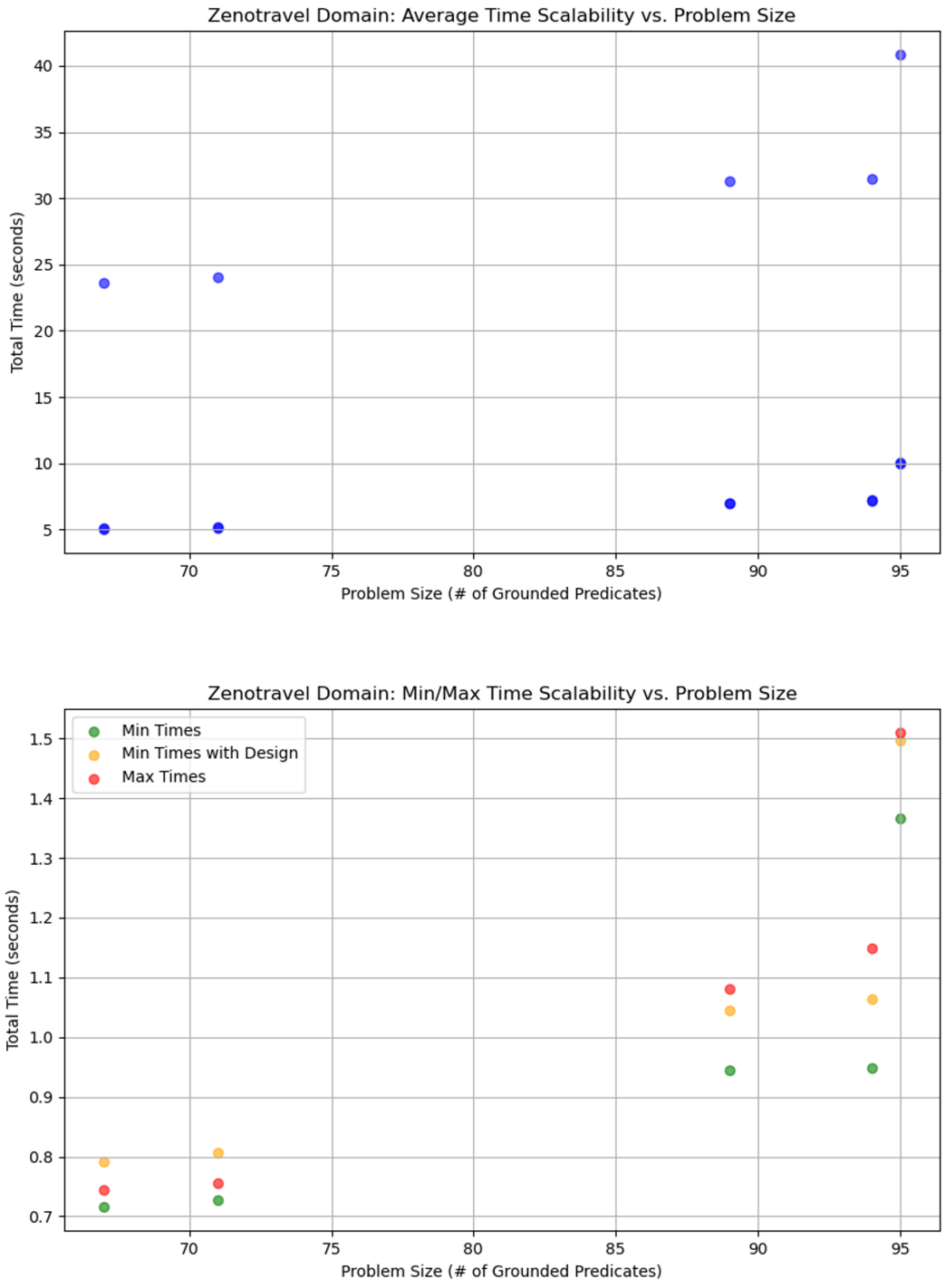
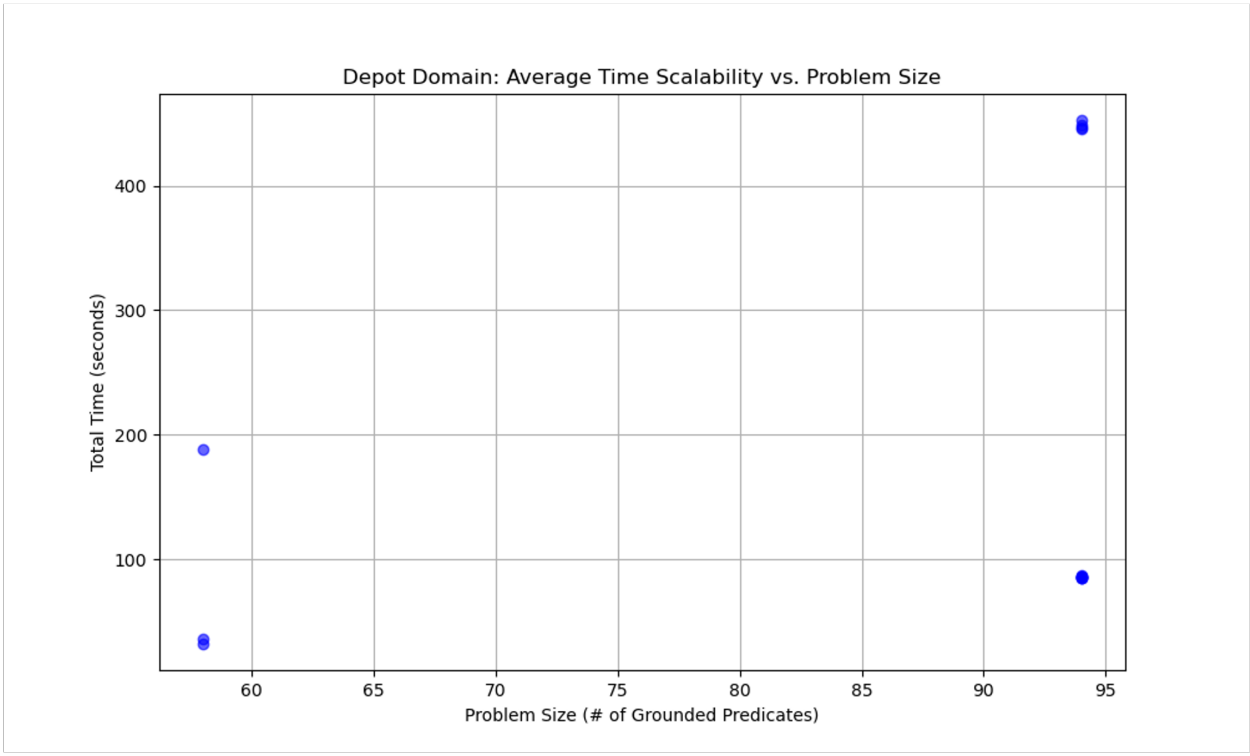


Figure A.1: A plot of the time taken for different problem sizes in Zenotravel.



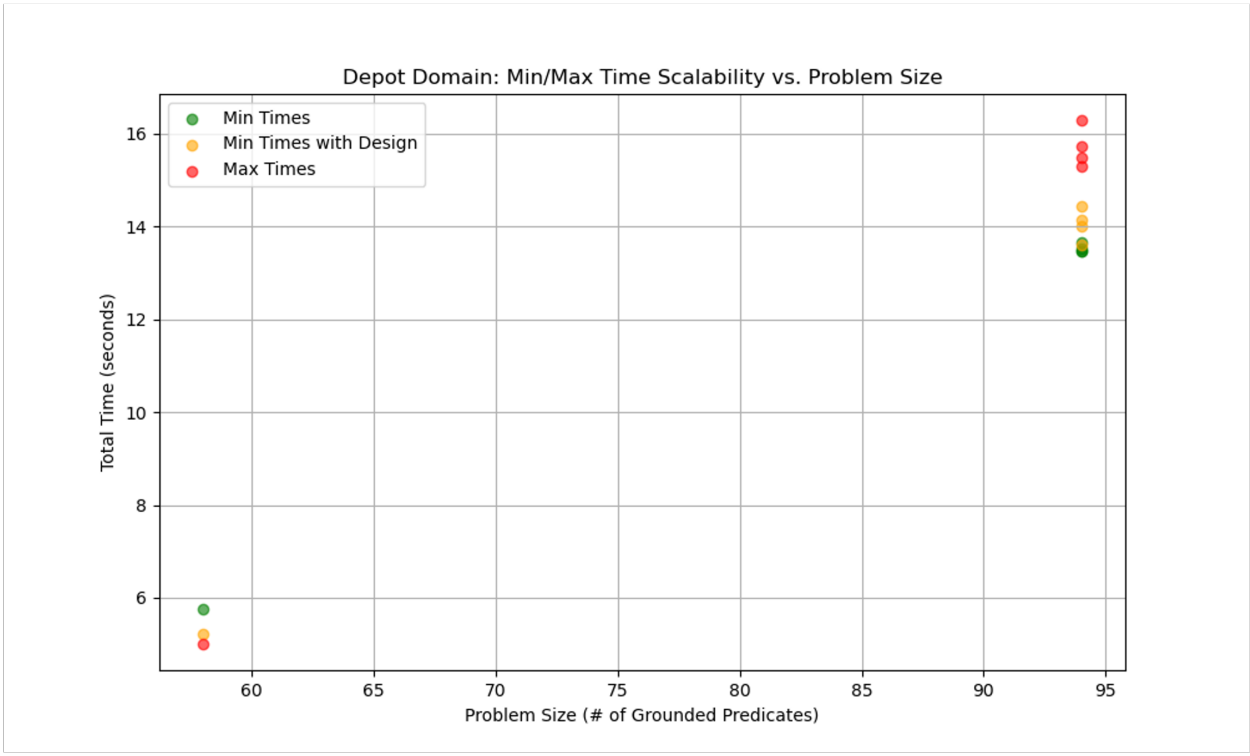


Figure A.2: A plot of the time taken for different problem sizes in Depot.

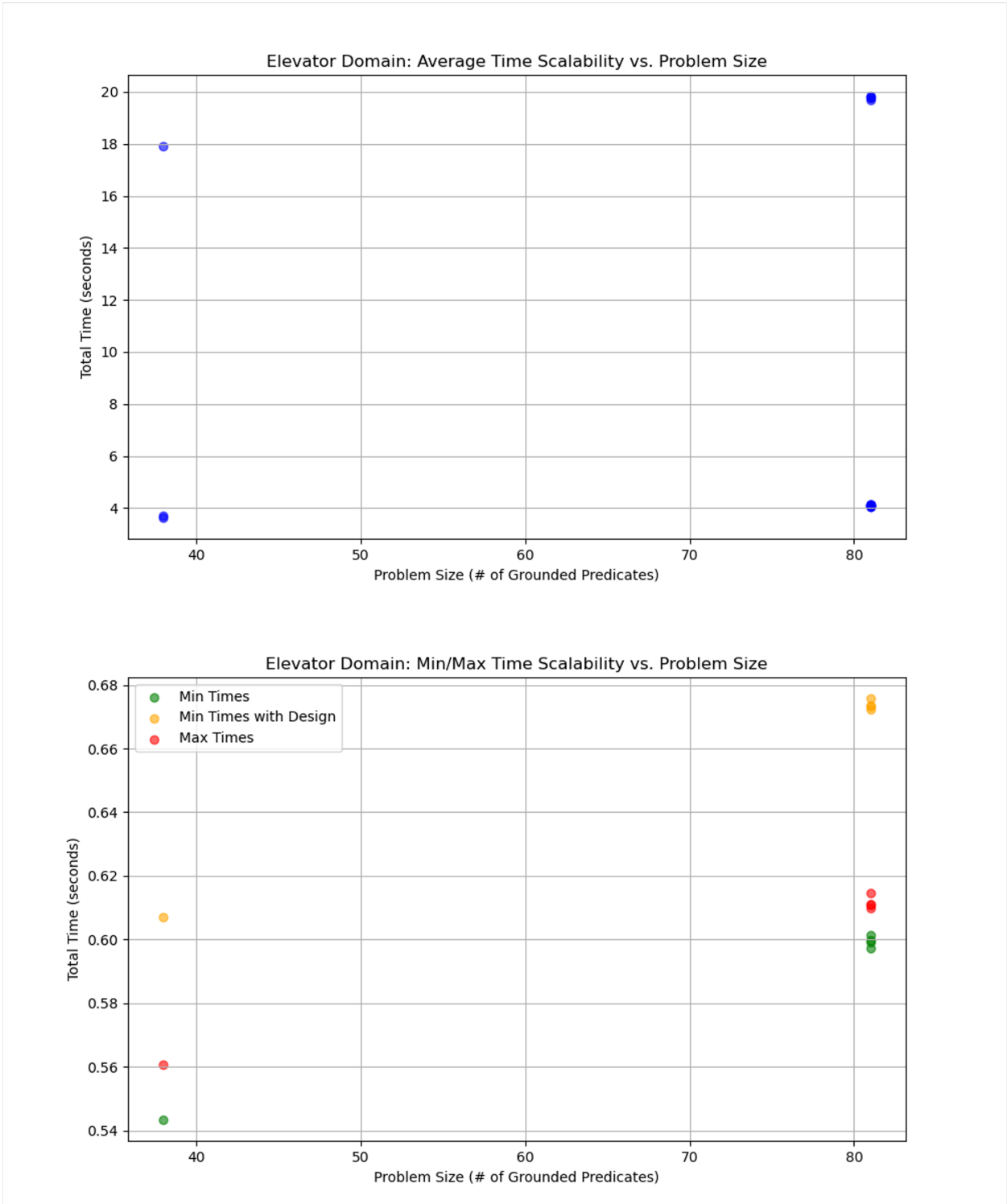


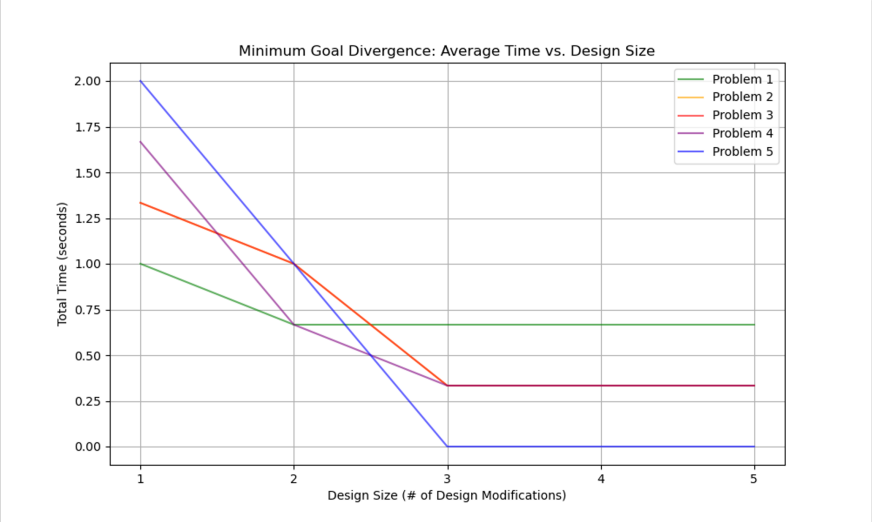
Figure A.3: A plot of the time taken for different problem sizes in Elevator.

Domain	Problem	Number of Grounded Predicates
Logistics	problem1	62
Logistics	problem2	62
Logistics	problem3	62
Logistics	problem4	62
Logistics	problem5	62
Block	problem1	71
Block	problem2	71
Block	problem3	89
Block	problem4	89
Block	problem5	89
Elev	problem1	38
Elev	problem2	81
Elev	problem3	81
Elev	problem4	81
Elev	problem5	81
Zeno	problem1	67
Zeno	problem2	71
Zeno	problem3	89
Zeno	problem4	94
Zeno	problem5	95
Depot	problem1	58
Depot	problem2	94
Depot	problem3	94
Depot	problem4	94
Depot	problem5	94

Table A.1: The total grounded predicate count per problem instance

A.2.2 Min and Max GSD Found for Different Design Budget

Next, we were interested in noting how the best case $\mathcal{GD}^{\downarrow}$ and $\mathcal{GD}^{\uparrow}$ evolved as we allowed for larger budgets. In this case, rather than creating problems where we were guaranteed designs that resulted in no goal-state divergence, we wanted to create problems with more flexibility. Here, we still created a human and robot problem copy, by random modifications, but one where we allowed both addition and removal of initial state fluents. We additionally performed a check to see if the problem resulting from the modifications was, in fact, solvable. We focused on a single domain, namely, the Elevator, and created three copies per instance. We then slowly increased the allowed domain modifications from one to five. Then, we noted



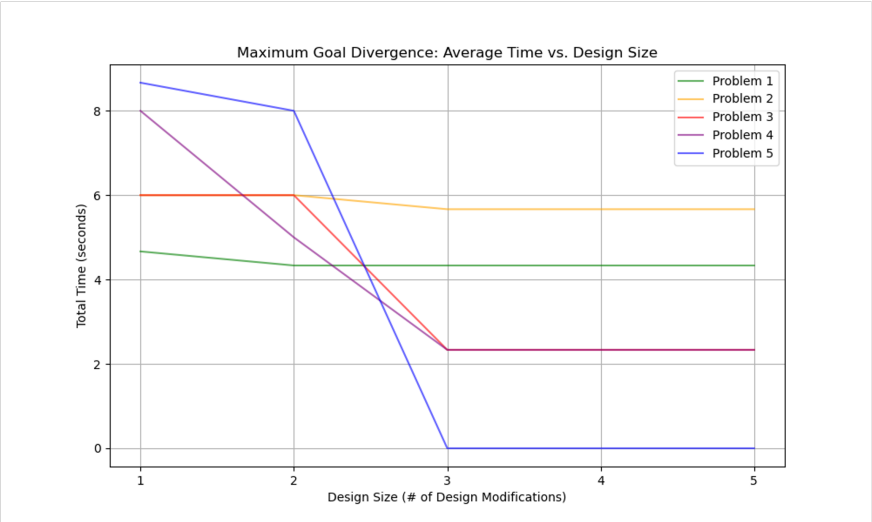


Figure A.4: A plot visualizing how the best $\mathcal{GSD}$ approximations change with design budget.

the best average minimum and maximum divergence we can accomplish within that budget (Figure A.4). As expected, we see increasing the budget does result in lower $\mathcal{GD}^{\downarrow}$ and $\mathcal{GD}^{\uparrow}$. We even see that for one of the problems, we were able to achieve zero $\mathcal{GD}^{\downarrow}$ and $\mathcal{GD}^{\uparrow}$.