Reducing Human-Robot Goal State Divergence with Environment Design

Kelsey Sikes,[1] **Sarah Keren,**[2] **Sarath Sreedharan,**[1]

[1]Department of Computer Science, Colorado State University, CO, USA
[2]The Taub Faculty of Computer Science, Technion-Israel Institute of Technology, Haifa, Israel
ksikes@colostate.edu, sarahk@cs.technion.ac.il, sarath.sreedharan@colostate.edu

Abstract

One of the most difficult challenges in creating successful human-AI collaborations is aligning a robot's behavior with a human user's expectations. When this fails to occur, a robot may misinterpret their specified goals, prompting it to perform actions with unanticipated, potentially dangerous side effects. To avoid this, we propose a new metric we call Goal State Divergence ($\mathcal{GSD}$), which represents the difference between a robot's final goal state and the one a human user expected. In cases where $\mathcal{GSD}$ cannot be directly calculated, we show how it can be approximated using maximal and minimal bounds. We then input the $\mathcal{GSD}$ value into our novel human-robot goal alignment (HRGA) design problem, which identifies a minimal set of environment modifications that can prevent mismatches like this. To show the effectiveness of $\mathcal{GSD}$ for reducing differences between human-robot goal states, we empirically evaluate our approach on several standard benchmarks.

Introduction

As Artificial Intelligence (AI) continues to advance and become a more ubiquitous part of society, human-robot interactions are becoming increasingly common. As a result, designing robots that exhibit behavior that conforms to human expectations is becoming more important than ever. Previous work (cf. (Zhang et al. 2017; Chakraborti et al. 2017)) has shown how addressing expectation mismatches lies at the heart of many human-AI interaction problems. In this paper, we will look at problems that might arise when there are differences between the potential goal states a human user expects a robot to achieve and those it might achieve.

Specifically, when the user provides a goal specification, they would have some expectation of the exact goal states that might satisfy them. However, the behavior the robot may choose in response to such a goal specification may result in a state that differs significantly from what the user expected in the characteristics not strictly provided in their specification. This in turn may result in unanticipated side effects, which in severe cases, could threaten human safety. Such expectation mismatches may arise for diverse reasons, including the human user misunderstanding the robot's state and capabilities or even limitations in their inferential capabilities. In this paper, we explore how environment design (Zhang and Parkes 2008) can be used to avoid such potential expectation mismatches. In particular, we look for ways to modify the environment to ensure that the difference between what the human user expects the robot to achieve and what the robot truly achieves is minimized for a given goal specification. We do so by driving the design process to minimize a novel metric called Goal State Divergence ($\mathcal{GSD}$), which identifies the discrepancy between the final goal state expected by the human and what can be achieved by the robot.

However, even under generous assumptions about the knowledge the designer has access to, estimating true $\mathcal{GSD}$ presents a unique challenge because the actual human plan may be unknown. In this paper, we instead aim to approximate the magnitude of the divergence through bounds and identify potential environmental modifications that can minimize it. Our paper also introduces novel classical planning-based compilations that can identify these bounds for a given design problem. To summarize, the primary contributions of this paper are as follows

1. We introduce a novel metric to characterize discrepancies between human-robot goal states in a given planning problem.
2. We develop approximations of the given metric and show how they can be effectively calculated using compilations to classical planning.
3. We introduce a novel design problem that leverages these approximations to minimize potential final goal state discrepancies.
4. We present a comprehensive empirical evaluation of our proposed method on standard benchmarks.

Related Work

Environment design shapes a robot's actions by modifying its environment to maximize or minimize some objective (Zhang and Parkes 2008; Keren, Gal, and Karpas 2019). Several early works in utilizing design in settings where the robots correspond to planning agents have focused primarily on using them to facilitate better goal and plan recognition (Keren, Gal, and Karpas 2014; Mirsky et al. 2019). Many of these works have relied primarily on heuristic search methods to identify such designs. (Keren et al. 2017a) looked at using design to maximize robot objectives in uncertain, stochastic environments, and (Keren et al. 2017b) leveraged

it to find the maximum shared agent-designer utility in Equi-Reward Utility Maximizing Design (ER-UMD) settings. For the latter, (Keren et al. 2021) extends this work by limiting the space of possible modifications, then mapping each one to a dominating modification. This avoids having to calculate all possible modifications. In our work, we propose a similar, method by identifying a bounded subset of modifications that meets certain criteria regarding the bounds on $\mathcal{GSD}$.

Many works have looked at approaches like value alignment (Hadfield-Menell et al. 2016; Mechergui and Sreedharan 2023) and avoiding side effects (Amodei et al. 2016; Weld and Etzioni 1994; Leike et al. 2017; Klassen, Alamdari, and McIlraith 2023; Klassen et al. 2022) as a means of ensuring safe behavior (the implicit assumption being that any behavior that avoids a certain set of states corresponding to the side-effect, will not cause any harm). Many of these works either assume access to a set of locked features or rely on directly querying the user to identify these features (cf. (Zhang, Durfee, and Singh 2018)). Methods like those proposed by (Saisubramanian and Zilberstein 2021) directly ask humans to update the environment to avoid negative side effects. Unfortunately, this method is hindered by the extensive human intervention it requires. Our method avoids such direct querying by instead relying on a specification of the human model to select an environment modification without requiring any further human intervention. We can learn such models by leveraging existing work on learning human mental models (cf. (Sreedharan et al. 2019)), in addition to all the works in learning planning models, in general (Callanan et al. 2022). Once the human domain knowledge is learned, it can be reused for multiple tasks. Additionally, in many cases, a set of people may share the same model, and we don't necessarily need to learn a unique model for each user (Soni, Sreedharan, and Kambhampati 2021).

Another related area of research is that of explicable planning (Zhang et al. 2017; Kulkarni et al. 2019), where a robot tries to generate plans aligned with a human's expectations about what plans the robot may choose. Recently, explicable planning has also been used to mitigate safety issues caused by human-AI model mismatches (Hanni, Boateng, and Zhang 2023), where a designer-specified safety bound is used to guarantee that an agent will never select an unsafe behavior. Environment design has also been applied to boost the ability of robots to generate explicable planning (cf. (Kulkarni et al. 2020)). Note that all the previously mentioned explicable planning methods generally focus on matching the human's expectations about the plan as a whole with the final plan carried out by the robot. On the other hand, we solely concentrate on matching the robot's final goal state with the human's expectations about potential final goal states and ignore the actual plans that may be used by the robot or expected by the human to achieve them.

Background

In this section, we define the basic planning terminologies we will be using throughout the paper. We define a planning model using the tuple $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$. Here $\mathcal{D}$ corresponds to the domain associated with the model and is further defined by the tuple $\mathcal{D} = \langle \mathcal{F}, \mathcal{A} \rangle$. $\mathcal{F}$ corresponds to the set of propositional fluents that describes the state space corresponding to the given planning problem, such that any state s in that space can be uniquely represented by the set of fluents that are true (i.e. $s \subseteq \mathcal{F}$, for all states s). $\mathcal{A}$ is a set of executable robot actions represented as the tuple $a = \langle pre_+(a), pre_-(a), add(a), del(a) \rangle$. For each action $a \in \mathcal{A}$, $pre_{+/-}(a) \subseteq \mathcal{F}$ are the set of positive or negative preconditions that must be satisfied before a can be executed, while $add(a)$ and $del(a)$ represent sets of add and delete effects for each action a. c corresponds to the cost associated with each action. Finally, $\mathcal{I}$ is the initial state, and $\mathcal{G} \subseteq F$ is the goal specification (which is a partial state specification and not necessarily a state). We define the effects of executing an action at a given state using a transition function $\mathcal{T}_\mathcal{M} : 2^\mathcal{F} \times A \rightarrow 2^\mathcal{F}$, which is given by

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

where $exec(s,a)$ returns true if $s \supseteq pre_+(a)$ and $s \not\supseteq pre_-(a)$. We will overload the notation and use the transition function to also be applicable to action sequences, such that $\mathcal{T}_\mathcal{M}(s, \langle a_1, ..., a_k \rangle) = \mathcal{T}_\mathcal{M}(...(\mathcal{T}_\mathcal{M}(s, a_1), ..., a_k))$.

A solution to a planning problem is a plan, which is an action sequence whose execution in the initial state results in a state that satisfies the goal specification, i.e., π is a plan if $\mathcal{T}_\mathcal{M}(\mathcal{I}, \pi) \supseteq \mathcal{G}$. We will refer to a state that satisfies the goal specification as a goal state. Each action in this plan has a cost; summing these reveals the cost of a plan, denoted by $c(\pi) = \sum_{a_i \in \pi} c(a_i)$. A plan is considered optimal if there exists no other plan with a lower cost, and we will represent the set of optimal plans for a model $\mathcal{M}$, with the notation $\Pi^*_\mathcal{M}$ and use $\Pi_\mathcal{M}$ to denote the set of all plans.

Running Example

Consider a robot operating in a greenhouse, tasked with completing various chores to maintain its operation. Here, a human assigns tasks to the robot based on their beliefs about its current state and capabilities. The robot then seeks to accomplish these tasks by following a plan that it believes will achieve the specified objective. In the best-case scenario, this plan may result in a goal state which is perfectly aligned with what the human was expecting or just lead to a few minor inconveniences. However, in the worst-case scenario, the robot could carry out a plan with potentially dangerous effects that the human did not anticipate.

As a specific example, consider a scenario where a human asks a robot to water a section of plants, sitting under a series of heat lamps. In providing this goal specification, the human expects the robot to carefully use a watering pail to water the plants, ensuring they remain adequately hydrated. Instead, the robot grabs a nearby hose and haphazardly sprays the plants, splashing water all over — including onto the heat lamps. This sudden change in temperature causes thermal shock, resulting in the heat lamps shattering and releasing sparks onto the plants below. Moments later, the plants ignite, setting the greenhouse on fire. The human who was not aware that the robot could use hoses completely overlooked this possibility.

To avoid situations like this, environment design can be a useful tool for influencing a robot's decision-making. In the greenhouse setting, for example, the robot's choice of what tool to water the plants with could have been dictated by their placement. Here, the watering pail could have been placed closer to the robot — increasing the possibility it'd be used — whereas the hose could have been placed farther away or left in an inaccessible position. Additionally, the heat lamps could have been outfitted with protective covers to ensure they remained shielded from any direct contact with water while the plants were being cared for. Our objective through this paper will be to design algorithms that can automatically identify such potential designs and limit potential mismatches.

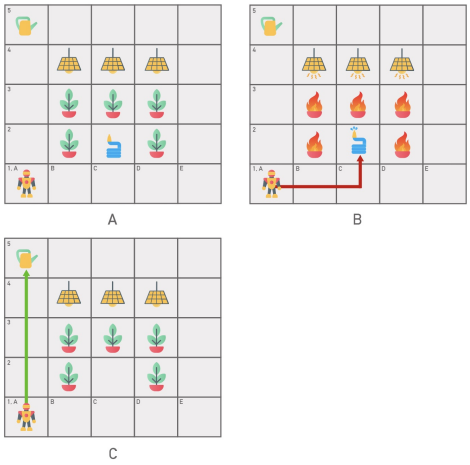

Figure 1: In a greenhouse setting, a human asks a robot to water plants based on their incorrect beliefs about its model. As a result, the robot follows the least costliest plan and chooses to water the plants with a hose, causing a fire. Using environment design, the hose is removed from the scene to avoid potential safety issues.

Design to Reduce Goal State Divergence

As discussed, the mismatch between the human's perception of the robot's capabilities/state and the reality could lead to users misspecifying their objective, potentially resulting in unanticipated outcomes. To develop methods that can account for and avoid such unintended outcomes, we first need to develop metrics to quantify the degree of mismatch. Specifically, we will start by looking at pairs of states.

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

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

In this paper, we are not interested in just measuring the difference between two arbitrary states but rather the goal state expected by the human and the goal state that the robot can achieve. One could make the case that the intermediate states the robot passes through are as important as the final goal state for many safety applications. However, it is important to note that a purely goal-based specification is general enough to account for such considerations easily. We can introduce new fluents that track intermediate states, and their value in the goal state can be used to account for whether the robot visited any undesirable intermediate state. This requires us to measure the difference in states achievable 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.:*

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

In our setting, these two models correspond to the robot model $\mathcal{M}^{\mathcal{R}} = \langle \mathcal{D}^{\mathcal{R}}, \mathcal{I}^{\mathcal{R}}, \mathcal{G}^{\mathcal{R}} \rangle$, and the human's model of the robot and the task $\mathcal{M}^{\mathcal{H}} = \langle \mathcal{D}^{\mathcal{H}}, \mathcal{I}^{\mathcal{H}}, \mathcal{G}^{\mathcal{H}} \rangle$. We are specifically looking at cases where the robot is trying to follow the goal specification provided by the human exactly, and thus, we have $\mathcal{G}^{\mathcal{H}} = \mathcal{G}^{\mathcal{R}}$. To simplify the notations, we will also assume that the human and robot models share the same fluent set $\mathcal{F}$. Let $\pi^{\mathcal{R}}$ be the robot plan and $\pi^{\mathcal{H}}$ be the plan expected by the human. The central metric of interest for this paper then becomes $\mathcal{GSD}(\pi^{\mathcal{H}}, \mathcal{M}^{\mathcal{H}}, \pi^{\mathcal{R}}, \mathcal{M}^{\mathcal{R}})$.

Note that calculating the above difference requires the system to have access to the human model $\mathcal{M}^{\mathcal{H}}$ and plan $\pi^{\mathcal{H}}$. As discussed, there are model learning methods we could employ to learn $\mathcal{M}^{\mathcal{H}}$; Additionally, we will assume that the human's model is given because under many structured settings, the human model may be known beforehand. In the greenhouse case, if the human has been working with a previous model of the robot, their beliefs about the robot would be heavily influenced by the model's capabilities.

It is worth noting that access to a human model doesn't mean that the robot could potentially avoid goal state divergence by executing plans that are valid in both models since such a plan might not exist. Coming to $\pi^{\mathcal{H}}$, even with a known $\mathcal{M}^{\mathcal{H}}$, the exact plan the human chooses may not be known beforehand because multiple plans may satisfy a goal state, any of which the human could choose.

In cases in which it is not possible to compute $\mathcal{GSD}$ exactly, we will instead consider approximations. The first approximation we will consider is the worst-case approximation, where we will look at the maximum divergence possible between an expected human goal state and what the robot can achieve, more formally,

Definition 3. *For two given models, $\mathcal{M}^1$, $\mathcal{M}^2$, the worst-case or maximal goal state 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 $\mathcal{M}^2$, 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 brings us to our first proposition which states that $\mathcal{GD}^{\uparrow}$ is guaranteed to be an upper bound of the true goal state divergence.

[1] The symmetric difference between two states is the number of elements present in either state but not both, which we denote using Δ.

Proposition 1. *For the robot and human model pair $\mathcal{M}^{\mathcal{R}}$ and $\mathcal{M}^{\mathcal{H}}$, the maximal goal state 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., $\mathcal{GD}^{\uparrow}(\mathcal{M}^{\mathcal{H}}, \mathcal{M}^{\mathcal{R}}) \geq |\mathcal{GSD}(\pi^{\mathcal{H}}, \mathcal{M}^{\mathcal{H}}, \pi^{\mathcal{R}}, \mathcal{M}^{\mathcal{R}})|$.*

The validity of the above proposition can be trivially proven from the definition of $\mathcal{GD}^{\uparrow}$. From the proposition, we can assert that one way to reduce goal state divergence is to reduce $\mathcal{GD}^{\uparrow}$. Especially, if we can reduce $\mathcal{GD}^{\uparrow}$ to zero, we are guaranteed that $\mathcal{GSD}$, will be an empty set.

However, $\mathcal{GD}^{\uparrow}$ could be a loose upper bound, and reducing $\mathcal{GD}^{\uparrow}$ will not necessarily always reduce $\mathcal{GSD}$. Another approximation we could use is the lower bound on $\mathcal{GSD}$. We define this measure similar to $\mathcal{GD}^{\uparrow}$, but now focusing on minimizing the divergence.

Definition 4. *For two given models, $\mathcal{M}^1$, $\mathcal{M}^2$, the best-case or minimal goal state 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 $\mathcal{M}^2$, i.e.:*

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

Similar to Proposition 1, we can assert that $\mathcal{GD}^{\downarrow}$ provides a lower bound.

Proposition 2. *For the robot and human model pair $\mathcal{M}^{\mathcal{R}}$ and $\mathcal{M}^{\mathcal{H}}$, the minimal goal state 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., $\mathcal{GD}^{\downarrow}(\mathcal{M}^{\mathcal{H}}, \mathcal{M}^{\mathcal{R}}) \leq |\mathcal{GSD}(\pi^{\mathcal{H}}, \mathcal{M}^{\mathcal{H}}, \pi^{\mathcal{R}}, \mathcal{M}^{\mathcal{R}})|$.*

We can characterize our unknown $\mathcal{GSD}$, using this upper bound, lower bound, and the gap between the two (i.e., $\mathcal{GD}^{\uparrow}(\mathcal{M}^{\mathcal{H}}, \mathcal{M}^{\mathcal{R}}) - \mathcal{GD}^{\downarrow}(\mathcal{M}^{\mathcal{H}}, \mathcal{M}^{\mathcal{R}})$).

Now with the basic setting and metrics in place, we are ready to finally define the design problem:

Definition 5. *A **human-robot goal-state alignment (HRGA)** design 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 or model updates. These may include changes to the state space, action preconditions, action effects, action costs, initial state, or goal.*
- *$\Lambda : \mathbb{M} \times \mathbb{U} \rightarrow \mathbb{M}$ 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.*

In the running example, model designs (we use the term design and environment modification interchangeably) may include moving objects like the watering pail or hose around the environment or removing them completely.

One could define various classes of solutions based on the metrics we have described above. The most basic one aims to minimize the design cost while requiring the lower bound and upper bound to fall below a specific threshold.

Definition 6. *A (l, k)-bounded minimal solution to a $\mathcal{DP}$ is the cheapest subset of modifications[2] ξ that satisfies the following conditions:*

$$\xi^* = \operatorname*{argmin}_{\xi \in 2^{\mathbb{U}}} \mathcal{C}(\xi)$$

Such that $\mathcal{GD}^{\downarrow}(\mathcal{M}^{\mathcal{R}}_{\xi}, \mathcal{M}^{\mathcal{H}}_{\xi}) \leq \ell$ and $\mathcal{GD}^{\uparrow}(\mathcal{M}^{\mathcal{R}}_{\xi}, \mathcal{M}^{\mathcal{H}}_{\xi}) \leq k$

Where $\mathcal{M}^{\mathcal{R}}_{\xi} = \Lambda(\mathcal{M}^{\mathcal{R}}, \xi)$ and $\mathcal{M}^{\mathcal{H}}_{\xi} = \Lambda(\mathcal{M}^{\mathcal{H}}, \xi)$

Though we focus on a single instance problem setting (i.e., design for a unique goal specification for an initial state), one could easily envision settings where the robot may be required to carry out a number of different tasks, each corresponding to a different goal specification. In such cases, the above definition can easily be extended to account for the multi-task nature of the setting. In particular, we can consider the max, min, or average of the $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$ values across instances. The specific variation that may be used might depend on the nature of the setting. For example, if one were to create designs to account for the worst possible case across all instances, one would want to make sure that the max of $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$ values across the instances fall below specific thresholds.

It is worth noting that the above definitions for the bounds consider a much larger set of plans than required. While the definitions consider all possible plans, humans may never consider most of those plans. For example, they might not think the robot would follow an extremely suboptimal plan. This will result in weaker bounds, which could result in more extensive model updates than required. In the running example discussed above, regardless of where you place the hose, there will always be a plan where the robot could go fetch the hose and spray the plants. Thus, if one were to make changes to the model based purely on these metrics, only removing the hose completely from the setting would result in a setting where the use of the hose is not considered part of one of the possible outcomes.

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

The first order of business for us would be to calculate the approximations of $\mathcal{GSD}$, in particular, we will show how one could employ an off-the-shelf cost-optimal planner to calculate these values. The general idea we will employ is the fact that we will create a single planning problem which involves coming up with actions in the robot model and human model, and then finally having a set of check actions that check how the goal state is achieved under the human plan and model compared against the one achieved under the robot plan and model.

More formally, given the model pair $\lambda = \langle \mathcal{M}^{\mathcal{R}}, \mathcal{M}^{\mathcal{H}} \rangle$, we create a new compiled model such that $\mathcal{M}^{\lambda} = \langle \mathcal{D}^{\lambda}, \mathcal{I}^{\lambda}, \mathcal{G}^{\lambda} \rangle$ where $\mathcal{D}^{\lambda}$ is the domain, defined by the tuple $\mathcal{D}^{\lambda} = \langle \mathcal{F}^{\lambda}, \mathcal{A}^{\lambda} \rangle$. Here, $\mathcal{F}^{\lambda}$ is a set of fluents represented by $\mathcal{F}^{\lambda} = \mathcal{F}^{\mathcal{R}} \cup \mathcal{F}^{\mathcal{H}} \cup \mathcal{F}^{\theta} \cup \mathcal{F}^{\kappa}$, where $\mathcal{F}^{\mathcal{R}}$ is the original set of fluents, and $\mathcal{F}^{\mathcal{H}}$ is a copy of these fluents which

[2]Note that this definition makes an implicit assumption that each design is independent and as such can be performed in any order.

correspond to the human's beliefs. We will use these copies to keep track of how the plan will unfold according to the human model and will use the notation $f_i^{\mathcal{H}}$ to denote the human copy of a fluent $f_i^{\mathcal{R}} \in \mathcal{F}^{\mathcal{R}}$. $\mathcal{F}^{\theta}$ includes the housekeeping fluents $robot\_can\_act$ and $human\_can\_act$ which control when a human or robot can perform actions, and $\mathcal{F}^{\kappa}$ are a set of compare fluents. $\mathcal{F}^{\kappa}$ contains a compare fluent for every fluent in $\mathcal{F}^{\mathcal{R}}$, i.e., $\exists f_i^{\kappa} \in \mathcal{F}^{\kappa}$, for every $f_i^{\mathcal{R}} \in \mathcal{F}^{\mathcal{R}}$. As discussed, our eventual objective is to compare the resultant goal states of the robot plan and a plan expected by the human. These compare fluents will allow us to track whether such comparisons have been performed.

$\mathcal{I}^{\lambda}$ is an initial state denoted by $\mathcal{I}^{\lambda} = \langle \mathcal{I}^{\mathcal{R}} \cup \mathcal{I}^{\mathcal{H}} \cup \{human\_can\_act\}\rangle$, where $\mathcal{I}^{\mathcal{R}}$ is the robot's initial state, and $\mathcal{I}^{\mathcal{H}}$ is a copy of this state, representative of the human's initial beliefs. The inclusion of $\{human\_can\_act\}$, ensures that the plan can start with human actions. $\mathcal{G}^{\lambda}$ is the set of goals shared by the human and robot, denoted by $\mathcal{G}^{\lambda} = \langle \mathcal{G}^{\mathcal{R}} \cup \mathcal{G}^{\mathcal{H}} \cup \mathcal{F}^{\kappa}\rangle$. Here, $\mathcal{G}^{\mathcal{R}} \subseteq \mathcal{F}^{\mathcal{R}}$ is the goal specified in the original fluent set, and $\mathcal{G}^{\mathcal{H}} \subseteq \mathcal{F}^{\mathcal{H}}$ the same goal expressed in the human fluent copy, while $\mathcal{F}^{\kappa}$ are the original compare fluents, used to determine how similar the human and robot's final goal states are, once all goals have been achieved.

$\mathcal{A}^{\lambda}$ is a set of actions represented by $\mathcal{A}^{\lambda} = \langle \mathcal{A}^{\mathcal{R}'} \cup \mathcal{A}^{\mathcal{H}'} \cup \mathcal{A}^{\theta} \cup \mathcal{A}^{\kappa}\rangle$. Here, $\mathcal{A}^{\mathcal{R}'}$ is the action set corresponding to the robot actions $\mathcal{A}^{\mathcal{R}}$. Here the action definitions are identical to their definitions in $\mathcal{A}^{\mathcal{R}}$, except that for any $a \in \mathcal{A}^{\mathcal{R}'}$, you have $robot\_can\_act \in pre(a)$. Similarly, $\mathcal{A}^{\mathcal{H}'}$ is a copy of these actions corresponding to the human's beliefs of them (i.e., corresponds to their definitions in $\mathcal{A}^{\mathcal{H}}$) expressed in $\mathcal{F}^{\mathcal{H}}$. Additionally, for any $a \in \mathcal{A}^{\mathcal{H}'}$, you have $human\_can\_act \in pre(a)$.

$\mathcal{A}^{\theta}$ are the special flip actions $a_{flip}^{\mathcal{R}}$ and $a_{flip}^{\mathcal{H}}$ which enable or disables a human or robot's ability to perform actions by changing the state of the $\mathcal{F}^{\theta}$ fluents. In our setting, the human begins with the ability to perform actions, while the robot does not. Once the human's goals have been achieved, their ability to perform actions is terminated, while the robot's are enabled. We define this specific human flip action, $pre_+(a_{flip}^{\mathcal{H}})$, as follows:

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

Once the robot has achieved all of its goals, its ability to perform actions is disabled using the action $a_{flip}^{\mathcal{R}}$. We can define $a_{flip}^{\mathcal{R}}$ similar to the $a_{flip}^{\mathcal{H}}$. These two actions ensure that the planner has identified a valid human and robot plan before performing all the check actions.

Once the human and robot have both executed their plans, all fluent sets from their final goal states are compared, for which one check fluent action exists for each fluent in $\mathcal{F}^{\mathcal{R}}$. $\mathcal{A}^{\kappa}$ is a set of compare actions that check for this consistency and is denoted by $\mathcal{A}^{\kappa} = A_{f_1}^{\kappa} \cup A_{f_2}^{\kappa} \cup A_{f_3}^{\kappa} ... A_{f_{|\mathcal{F}|}}^{\kappa}$, such that the set $\mathcal{A}_{f_i}^{\kappa} = \{a_{f_i}^1, a_{f_i}^2, a_{f_i}^3, a_{f_i}^4\}$ exists for each $f_i^{\mathcal{R}} \in \mathcal{F}^{\mathcal{R}}$. We will call the first two copies the check disagreement actions for fact f_i and the latter two the check agreement actions. The agreement copies will only fire if the human's belief about the fluent value matches the robot's, and the disagreement copy fires only in case they don't. Additionally, we will modulate the cost parameters $\mathcal{P}_1$ (agreement action cost) and $\mathcal{P}_2$ (disagreement action cost) to get different behaviors from the compilation. These are defined as follows:

- $pre_+(a_{f_i}^1) = \{f_i^{\mathcal{R}}\}$,
  $pre_-(a_{f_i}^1) = \{f_i^{\mathcal{H}}\} \cup \{robot\_can\_act\} \cup \{human\_can\_act\} \cup \{f_i^{\kappa}\}$,
  $add(a_{f_i}^1) = \{f_i^{\kappa}\}$, $del(a_{f_i}^1) = \emptyset$, and $c(a_{f_i}^1) = \mathcal{P}_1$
- $pre_+(a_{f_i}^2) = \{f_i^{\mathcal{H}}\}$,
  $pre_-(a_{f_i}^2) = \{f_i^{\mathcal{R}}\} \cup \{robot\_can\_act\} \cup \{human\_can\_act\} \cup \{f_i^{\kappa}\}$,
  $add(a_{f_i}^2) = \{f_i^{\kappa}\}$, $del(a_{f_i}^2) = \emptyset$, and $c(a_{f_i}^2) = \mathcal{P}_1$
- $pre_+(a_{f_i}^3) = \{f_i^{\mathcal{R}}, f_i^{\mathcal{H}}\}$,
  $pre_-(a_{f_i}^3) = \{robot\_can\_act\} \cup \{human\_can\_act\} \cup \{f_i^{\kappa}\}$,
  $add(a_{f_i}^3) = \{f_i^{\kappa}\}$, $del(a_{f_i}^3) = \emptyset$, and $c(a_{f_i}^3) = \mathcal{P}_2$
- $pre_+(a_{f_i}^4) = \emptyset$,
  $pre_-(a_{f_i}^4) = \{f_i^{\mathcal{R}}, f_i^{\mathcal{H}}\} \cup \{robot\_can\_act\} \cup \{human\_can\_act\} \cup \{f_i^{\kappa}\}$,
  $add(a_{f_i}^4) = \{f_i^{\kappa}\}$, $del(a_{f_i}^4) = \emptyset$, and $c(a_{f_i}^4) = \mathcal{P}_2$

For a plan π^{λ} that is valid for this new model $\mathcal{M}^{\lambda}$, we will use the notation $\mathcal{H}(\pi^{\lambda})$ to represent the sequence of human actions that appear in π^{λ}, and $\mathcal{R}(\pi^{\lambda})$ to represent the robot actions. Also, we will use the notation $\kappa^+(\pi^{\lambda})$ and $\kappa^-(\pi^{\lambda})$, to list the set of check agreement and check disagreement actions that appear in the plan.

One of the aspects of the model definition we haven't delved into is the action costs of the different actions. As we will see setting these costs to different values allows us to determine the values we are interested in. In general, we will assume that the cost of all actions in $\mathcal{A}^{\theta}$ are zero.

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 $\mathcal{GD}^{\uparrow}(\mathcal{M}^{\mathcal{R}}, \mathcal{M}^{\mathcal{H}}) = |\kappa^-(\pi^{\lambda})|$.*

Proof. Now this comes from the fact that the cost of the plan is being dominated by check agreement actions. In particular, the cost of a single agreement action is higher than the combined cost of the longest possible plan in either the human or robot model (i.e., one that passes through each possible state). By setting the cost of agreement so high, we force the planner to select plans with a low degree of agreement. □

We can similarly calculate the $\mathcal{GD}^{\downarrow}(\mathcal{M}^{\mathcal{R}}, \mathcal{M}^{\mathcal{H}})$, by inverting the costs, specifically:

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 proof is identical to the previous proposition.

Remark One of the additional constraints we are placing on solutions to this problem is the requirement that human actions be performed before any robot actions. This is technically not a requirement for the validity of the compilation. If we had added $\{robot\_can\_act\}$ to the initial state, the compilation will allow for human and robot actions to be interleaved or picked in any order. We will refer to this version as the flattened version of the compilation. Flattening the compilation will allow for more solutions but at the cost of increasing the branching factor. One of the evaluations, we will perform is whether the two versions have any significant difference in computational characteristics.

Identifying Minimal Designs for HRGA

Now that we have methods for computing the $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$ bounds, the question remains as to how to select the designs that will allow us to create models with the required properties for a given HRGA $\mathcal{DP}$. In particular, we are interested in identifying designs that meet the requirements laid out in Definition 6. However, rather than laying out the most general version, we will look at a specific instantiation of the definition that will allow us to use an even more efficient compilation than the version laid out in the previous section. In particular, we are interested in settings, where $\ell = 0$ (the limit on the lower bound) and the set of design changes provided as input, correspond to adding or removing unique fluents from the human/robot initial states and each design has a unit cost.

Here, we only allow initial state changes because, for most practical problem settings, initial state changes are the easiest changes the designer could make. From a theoretical point of view, one could always map changes to any other model component into an initial state change. For example, by making them conditioned on a static predicate, whose value is determined in the initial state.

The basic algorithm will have two loops, the outer loop will iteratively increase the allowed design cost. The inner loop will try to identify a design for the given budget constraint that will meet the requirement of $\mathcal{GD}^{\downarrow}$ of zero, and $\mathcal{GD}^{\uparrow}$ within some specified limit.

Inner Loop for Identifying Designs

In the inner loop, we will follow a slightly modified version of $\mathcal{GD}^{\downarrow}$ to identify the design itself. For a set of possible designs $\mathbb{U}$, we can map each design to a specific addition or removal to the initial state for the human and robot model. Let τ be the current limit placed on the design size. The basic intuition here is that we will modify the $\mathcal{GD}^{\downarrow}$ compilation to first perform a set of actions corresponding to design changes. The design actions are disabled to calculate the human and robot plan that results in $\mathcal{GD}^{\downarrow}$ with 0. We can directly encode this into the goal by looking for plans where all the fluent values match. We check whether the identified design allows the required k bound on the $\mathcal{GD}^{\uparrow}$. If not,

Algorithm 1: An algorithm for a HRGA design problem

Input: $\mathcal{DP}, k$
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 k
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 k **then**
        return $\mathcal{U}$
      end if
    end if
  end while
end for

we look for another design with τ length which satisfies the $\mathcal{GD}^{\downarrow} = 0$ requirement. We do this by updating the $\mathcal{GD}^{\downarrow}$ compilation to disallow previously identified designs.

We will extend our previous compiled model as follows, $\mathcal{M}^{\lambda}$ to $\mathcal{M}^{\lambda}_{\mathbb{U}} = \langle \mathcal{D}^{\lambda}_{\mathbb{U}}, \mathcal{I}^{\lambda}_{\mathbb{U}}, \mathcal{G}^{\lambda}_{\mathbb{U}} \cup \{unseen\_design\}\rangle$ where $\mathcal{D}^{\lambda}_{\mathbb{U}} = \langle \mathcal{F}^{\lambda}_{\mathbb{U}}, \mathcal{A}^{\lambda}_{\mathbb{U}}, c^{\lambda}_{\mathbb{U}}\rangle$. In this new model, we have $\mathcal{F}^{\lambda}_{\mathbb{U}} = \langle \mathcal{F}^{\lambda} \cup \{design\_allowed\} \cup \{unseen\_design\} \cup \mathcal{F}^{\tau} \cup \mathcal{F}^{\tau+} \cup \mathcal{F}^{\mathcal{D}}\rangle$. Where $\{design\_allowed\}$ is used to keep track of when designs are allowed and $\{unseen\_design\}$ ensures that the current design used hasn't been used before. The fluent sets $\mathcal{F}^{\tau}$ and $\mathcal{F}^{\tau+}$ ensures only τ designs can be performed. Finally, $|\mathcal{F}^{\mathcal{D}}| = |\mathbb{U}|$ keeps track of what exact designs were used.

For the actions we have, $\mathcal{A}^{\lambda}_{\mathbb{U}} = \langle \mathcal{A}^{\lambda} \cup \mathcal{A}^{\mathbb{U}} \setminus \mathcal{A}^{\kappa-} \cup \{design\_completed\}\rangle$. Where $|\mathcal{A}^{\mathbb{U}}| = \tau \times |\mathbb{U}|$, is the set of actions you have for the design, and $\mathcal{A}^{\kappa-}$ is the subset of check disagreement copies. Each design action updates the initial state per the design requirements. $\{design\_completed\}$ stops the design phase and allows the human and robot actions to be applied (from there on out, the actions are the same as the previous compilation). By not including the disagreement copy, we will look for robot/human plan pairs that can only satisfy the original check goal by using agreement actions (hence, the states need to match).

The new initial state is given as $\mathcal{I}^{\lambda}_{\mathbb{U}} = (\mathcal{I}^{\lambda} \setminus \{human\_can\_act\}) \cup \{unseen\_design, design\_allowed\} \cup \mathcal{F}^{\tau}$. Therefore, you can only start with design steps (which can be performed at most k steps).

For the new design action, there exists an action for each possible design step (upper-bounded by k) and a design. For a design related to fluent f and step i, the positive precondition of the action would be $pre_{+}(a) = \{design\_allowed, k_i\}$. If the design corresponds to making an initial state true, that fluent is part of the add effect, if makes a fluent false it becomes part of the delete effect.

The action will always remove $t_i \in \mathcal{F}^\tau$ and add $t_i^+ \in \mathcal{F}^{\tau+}$ as well as the corresponding design. Now the goal is given as $\mathcal{G}^\lambda_\mathbb{U} = \langle \mathcal{G}^\lambda \cup \mathcal{F}^{\tau+} \rangle$. The $\{design\_completed\}$ action simply deletes $\{design\_allowed\}$ and adds the fluent $\{human\_can\_act\}$. Now the addition of $\mathcal{F}^{\tau+}$ means that τ design needs to be applied. The cost function is kept the same as $\mathcal{M}^\lambda$. A solution to this problem allows us to identify designs that result in zero $\mathcal{GD}^\downarrow$ automatically, and we can subsequently check $\mathcal{GD}^\uparrow$. If $\mathcal{GD}^\uparrow$ requirements are met, we know that this corresponds to the minimal cost design, and the solution is returned.

If this is not the case, we would want to disallow it and look for other designs of size τ that might suit our requirements. We will do this by introducing new conditional effects into the $\{design\_completed\}$ action, such that the condition for that effect corresponds to the design fluents of a previously identified design and the effect is to delete the $\{design\_completed\}$ fluent.

Once the updated $\mathcal{M}^\lambda_\mathbb{U}$ no longer returns a solution, we know that no other minimal designs of that budget satisfy this requirement and the control is passed to the outer loop for checking a larger design budget. Algorithm 1 provides a pseudo-code for this algorithm. $found\_designs$ is a set that is used to track all the previously found designs for the current design budget, and $all\_designs\_found$ is a flag that captures whether the algorithm has exhausted the space of all designs that can ensure a $\mathcal{GD}^\downarrow$ of zero.

Evaluation

Our empirical evaluation objective was to provide a computational characterization of the different approaches to computing $\mathcal{GD}^\uparrow$ and $\mathcal{GD}^\downarrow$ measures and to perform designs that were introduced in this paper.

Dataset We looked at five standard IPC domains (IPC 2016a,b), and converted five problem instances from each into variations of a goal state divergence problem. To help minimize our problem run-time and potential planner issues, instances with smaller initial states were chosen.

To create the human and robot models, these instances were first duplicated. For each instance, five problem variations containing human and robot models were then created. All original robot problem instances were kept the same as the original IPC problem instance. We created the human problem instance by deleting five random initial state fluents from the initial state of the original instance. This means five problem variations for each problem instance for each domain were created, for a total of 25 problem variations created per domain. All values listed in the table are averaged across these five randomly generated problem variations. This ensured that there was always a design set of size five within the required $\mathcal{GD}^\uparrow$ and $\mathcal{GD}^\downarrow$ limits. We also removed the zoom action from Zenotravel to avoid large variations in the fuel level fluents.

For consistency, we considered an $\mathcal{GD}^\uparrow$ limit of 0 as well. This allowed us to frame the calculation $\mathcal{GD}^\uparrow$ for the problem as checking whether the $\mathcal{GD}^\uparrow$ compilation is unsolvable if we force the plan to have at least one disagreement action. This allows us to perform cross-domain comparisons while keeping the $\mathcal{GD}^\uparrow$ constant and also avoid the use of costlier cost-optimal planners.

To guarantee that we had problems with the required $\mathcal{GD}^\uparrow$ limit, we updated the goal specification of the problem instances selected from previous IPC competitions so all plans would result in the same goal state.

Setup We implemented the compilations for individually computing baselines of $\mathcal{GD}^\uparrow$ and $\mathcal{GD}^\downarrow$ as well as the updated $\mathcal{GD}^\downarrow$ compilation that also identifies the design. We also implement a simple breadth-first search over the design space for the baseline.

For design, our primary points of comparison will be our proposed algorithm (referred to as `Main`) and a naive one (`Naive`) that merely iterates over all possible designs and tests whether the designs result in zero upper and lower bounds. We will also consider a variation of the `Main` that considers the flattened compilation (`Main-fl`).

For each of these primary design algorithms (i.e., `Main`, `Main-fl`, and `Naive`), the time listed is the total time taken to find the minimal design that will ensure the upper and lower bounds are zero. As such, this involves solving for upper and lower bounds multiple times. Conversely, the times listed for $\mathcal{GD}^\uparrow$, $\mathcal{GD}^\downarrow$, and $\mathcal{GD}^\downarrow$ with design is the average time taken to compute each of these bounds individually (with ordering constraints enforced). To the best of our knowledge, we are the first to tackle this problem, and we are unaware of any existing baselines to compare this work against. Thus, we only consider baselines that provide the minimal design for the target upper and lower bounds but with potentially different computational overheads (we have also provided a characterization of the hardness of calculating these bounds).

For each domain, we tested the three conditions on each instance, and we used Lama (Richter and Westphal 2010) for solving all compilations. All experiments were performed on a computer with an Apple M2 Max chip and 64 GB Ram. All experiments were run with a time limit of 60 minutes.

Results Our primary metric is the time taken by each approach. Accordingly, Table 1 presents the average and standard deviation time in seconds taken per each instance reported. Across all problems, we see that our **`Main` and `Main-fl` methods take a significantly much shorter time than the baseline**. For, `Main` and `Main-fl` were mostly comparable, with small variation between instances. As such, we see that enforcing the ordering does provide a small improvement over the flattened compilation. Presumably, this is due to the fact that adding the additional structure would reduce the branching factor. Also, compared with the naive baseline, which does not leverage planning to identify the design, takes a significantly shorter time. We note that the most noticeable benefit is for the Depot domain. We also note that the addition of design into $\mathcal{GD}^\downarrow$ compilation adds minimal overhead. Finally, the $\mathcal{GD}^\uparrow$ times are, in general, higher than $\mathcal{GD}^\downarrow$ times. However, this is expected since, in this setting, $\mathcal{GD}^\uparrow$ corresponds to testing for unsolvability.

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 1: 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.

Conclusion

This paper presents the first attempt at developing a design framework to help align human expectations about how a goal specification may be achieved with the actual outcomes of a robot plan selected to satisfy the specification. Our focus in this paper has been to provide a clear framework to understand and study environment design within this context. As alluded to in the paper, the specific design problem we study is one among a number of different problems we could study in this space. In future work, we hope to explore some of these works and also look at studying these problems in the context of more complex decision-making frameworks. In particular, we would be interested in seeing how to adapt these mechanisms to support more complex objective/preference specification mechanisms including various forms of temporal logic and reward functions.

References

2016a. 2nd International Planning Competition, 2000.

2016b. 3rd International Planning Competition, 2002. https://github.com/potassco/pddl-instances/tree/master/ipc-2002.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Mané, D. 2016. Concrete Problems in AI Safety. *arXiv preprint arXiv:1606.06565*.

Callanan, E.; Venezia, R. D.; Armstrong, V.; Paredes, A.; Chakraborti, T.; and Muise, C. 2022. MACQ: A Holistic View of Model Acquisition Techniques. arXiv:2206.06530.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *IJCAI 2017*, 156–163. IJCAI Organization.

Hadfield-Menell, D.; Russell, S.; Abbeel, P.; and Dragan, A. D. 2016. Cooperative Inverse Reinforcement Learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 3909–3917. Barcelona, Spain: Curran Associates, Inc.

Hanni, A.; Boateng, A.; and Zhang, Y. 2023. Safe Explicable Robot Planning. arXiv:2304.03773.

Keren, S.; Gal, A.; and Karpas, E. 2014. Goal Recognition Design. In *Proceedings of the Twenty-Fourth International Conferenc on International Conference on Automated Planning and Scheduling*, ICAPS'14, 154–162. AAAI Press. ISBN 9781577356608.

Keren, S.; Gal, A.; and Karpas, E. 2019. Goal Recognition Design in Deterministic Environments. *J. Artif. Int. Res.*, 65: 209–269.

Keren, S.; Gal, A.; Karpas, E.; Pineda, L.; and Zilberstein, S. 2017a. Redesigning Stochastic Environments for Maxi-

mized Utility. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

Keren, S.; Pineda, L.; Gal, A.; Karpas, E.; and Zilberstein, S. 2017b. Equi-Reward Utility Maximizing Design in Stochastic Environments. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, 4353–4360. AAAI Press. ISBN 9780999241103.

Keren, S.; Pineda, L.; Gal, A.; Karpas, E.; and Zilberstein, S. 2021. Efficient Heuristic Search for Optimal Environment Redesign. *Proceedings of the International Conference on Automated Planning and Scheduling*, 29(1): 246–254.

Klassen, T. Q.; Alamdari, P. A.; and McIlraith, S. A. 2023. Epistemic Side Effects: An AI Safety Problem. In *Proceedings of the 2023 AAMAS*, 1797–1801.

Klassen, T. Q.; McIlraith, S. A.; Muise, C.; and Xu, J. 2022. Planning to avoid side effects. In *AAAI*, volume 36, 9830–9839.

Kulkarni, A.; Sreedharan, S.; Keren, S.; Chakraborti, T.; Smith, D. E.; and Kambhampati, S. 2020. Designing Environments Conducive to Interpretable Robot Behavior. In *2020 IROS*, 10982–10989.

Kulkarni, A.; Zha, Y.; Chakraborti, T.; Vadlamudi, S. G.; Zhang, Y.; and Kambhampati, S. 2019. Explicable Planning as Minimizing Distance from Expected Behavior. In *AAMAS*, AAMAS '19, 2075–2077. ISBN 9781450363099.

Leike, J.; Martic, M.; Krakovna, V.; Ortega, P. A.; Everitt, T.; Lefrancq, A.; Orseau, L.; and Legg, S. 2017. AI Safety Gridworlds. arXiv:1711.09883.

Mechergui, M.; and Sreedharan, S. 2023. Goal Alignment: Re-Analyzing Value Alignment Problems Using Human-Aware AI. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, 2331–2333. ISBN 9781450394321.

Mirsky, R.; Gal, K.; Stern, R.; and Kalech, M. 2019. Goal and Plan Recognition Design for Plan Libraries. *ACM Trans. Intell. Syst. Technol.*, 10(2).

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research*, 39: 127–177.

Saisubramanian, S.; and Zilberstein, S. 2021. Mitigating Negative Side Effects via Environment Shaping. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, 1640–1642. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450383073.

Soni, U.; Sreedharan, S.; and Kambhampati, S. 2021. Not all users are the same: Providing personalized explanations for sequential decision making problems. arXiv:2106.12207.

Sreedharan, S.; Hernandez, A. O.; Mishra, A. P.; and Kambhampati, S. 2019. Model-Free Model Reconciliation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, 587–594. AAAI Press. ISBN 9780999241141.

Weld, D.; and Etzioni, O. 1994. The First Law of Robotics (a Call to Arms). In *Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence*, AAAI'94, 1042–1047. AAAI Press.

Zhang, H.; and Parkes, D. 2008. Value-Based Policy Teaching with Active Indirect Elicitation. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 1*, AAAI'08, 208–214.

Zhang, S.; Durfee, E. H.; and Singh, S. 2018. 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*, IJCAI'18, 4867–4873. AAAI Press. ISBN 9780999241127.

Zhang, Y.; Sreedharan, S.; Kulkarni, A.; Chakraborti, T.; Zhuo, H. H.; and Kambhampati, S. 2017. Plan explicability and predictability for robot task planning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 1313–1320.