Reducing Goal State Divergence with Environment Design

Kelsey Sikes[1], **Sarah Keren**[2], **Sarath Sreedharan**[1]
[1]Colorado State University
[2]Technion-Israel Institute of Technology
ksikes@colostate.edu, sarahk@technion.ac.il, sarath.sreedharan@colostate.edu

Abstract

Generating behaviors that align with human expectations is a key requirement for human-robot collaboration. Potential behavior misalignment could lead to the robot performing actions with unanticipated, potentially dangerous side effects even while pursuing human goals. In this paper, we introduce a novel metric called Goal State Divergence ($\mathcal{GSD}$) which quantifies the difference between the state a robot achieved in response to a human-specified goal and what the human expected. In cases where $\mathcal{GSD}$ cannot be directly calculated, we show how it can be approximated using maximal and minimal bounds. We then leverage $\mathcal{GSD}$ in our novel human-robot goal alignment design (HRGAD) problem, which identifies a minimal set of environment modifications that can reduce such mismatches. We show the effectiveness of our method in reducing the goal state divergence by empirically evaluating our approach on several planning benchmarks.

Code — https://github.com/SikesK/GSD_AAAI

Introduction

Avoiding unanticipated side-effects and unexpected agent behavior in response to a legitimate human request is widely considered to be one of the central problems within AI safety (Saisubramanian and Zilberstein 2021; Sreedharan 2023). Such problems are further exacerbated in the context of robotics, where an everyday human may not completely understand a robot's capabilities and limitations. This means that the behavior a robot generates in response to a human-specified goal may differ significantly from what they expected, possibly resulting in outcomes that are not only unanticipated but potentially dangerous.

To illustrate this problem and a potential solution, consider a scenario where a robot is tasked with taking care of chores in a greenhouse (Fig. 1). Here, the human tasks the robot with periodically watering the plants, expecting the robot to use the watering pail. However, the robot grabs a nearby hose and haphazardly sprays water everywhere. The human never expected this course of action, as they were unaware that the robot could use the hose. While this plan achieves the specified goal, i.e., watering the plants, the robot action also has the unintended effect of wetting the heat lamps. The resulting thermal shock could cause the heat lamps to shatter, spreading sparks and burning down the entire greenhouse. However, this undesired outcome could be avoided by simply removing the hose from the greenhouse. The above example illustrates the application of a solution concept called environment design (Zhang and Parkes 2008). Under environment design, the planning problem itself is modified to promote or avoid certain agent behaviors, which in the above case involves removing the hose.

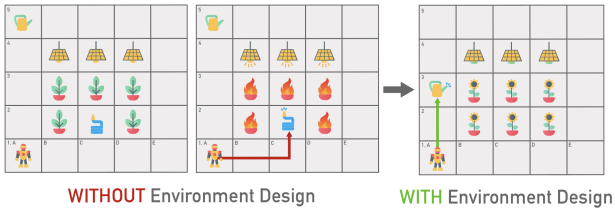

Figure 1: An illustration of the greenhouse example. The optimal plan here leads to the robot watering the plants with a hose, causing a fire. Using environment design, the hose is removed from the scene to avoid potential safety issues.

Here, we will propose a novel metric called Goal State Divergence ($\mathcal{GSD}$), which quantifies the discrepancy between the final goal state expected by the human and what can be achieved by the robot. As we will see, it might not always be possible to estimate the true $\mathcal{GSD}$ value. As such, we will develop methods to approximate this value through lower and upper bounds and identify potential environmental modifications that can minimize it. Specifically, we will leverage novel classical planning-based compilations to determine the bounds and potential environment designs that can minimize it. We then perform a comprehensive empirical evaluation of our proposed method on planning benchmarks to establish its computational characteristics.

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; Pozanco, Fraga Pereira, and Borrajo 2024). 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. Similarly, unsupervised environment design (Monette et al. 2025; Wang et al. 2019, 2020; Teoh, Li, and Varakantham 2025) also influences an agent's behavior by manipulating its environment. Here environments are automatically generated and modified to drive agent learning. These differ from our framework that aims at identifying modification with respect to a pre-defined criteria.

Many works have looked at approaches like value alignment (Hadfield-Menell et al. 2016; Mechergui and Sreedharan 2024; Sreedharan 2025; Gabriel and Ghazavi 2023; Sorensen et al. 2024; Ji et al. 2025) 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; Steinmetz et al. 2022) as a means of ensuring safe behavior. In the latter case, they assume that avoiding certain states corresponds to not causing any harm. Many of these works either assume access to locked features or rely on directly querying the human to identify these features (cf. (Zhang, Durfee, and Singh 2018)). Previous works have also tried directly asking humans to update the environment to avoid negative side effects (Saisubramanian and Zilberstein 2021). Unfortunately, this method is hindered by the extensive human intervention it requires. Our method avoids such direct querying by instead relying on an estimate of the human model to select an environment modification that can avoid unexpected behavior without needing further human intervention. We can learn such models by leveraging existing work on learning human mental models (Sreedharan et al. 2019; Reddy, Dragan, and Levine 2018), in addition to all the works in learning planning models (Callanan et al. 2022). The learned human model can then be reused for multiple tasks. Often, humans may share the same model, and we don't necessarily need to learn a unique model for each human (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 been used to mitigate safety issues caused by human-AI model mismatches (Hanni, Boateng, and Zhang 2024; Hanni, Montaño, and Zhang 2025), 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 explicable planning focuses on matching observed behavior while ignoring how far the final resultant state, i.e., the impact of executing the plan, may be from the expected one. Our work focuses on measuring the distance between the resultant and expected state.

Background

In this section, we define the basic planning terminologies we will be using throughout the paper. We consider a case where the robot state estimator returns a state representation that makes use of a set of boolean variables. Here, we represent the planning problem using a tuple of the form $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$. $\mathcal{D}$ corresponds to the domain associated with the model and is further defined by the tuple $\mathcal{D} = \langle \mathcal{F}, \mathcal{A} \rangle$. $\mathcal{F}$ is the set of boolean features, also referred to as propositional fluents or just fluents, that describes the state space of 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 $\mathcal{G} \subseteq \mathcal{F}$ is the goal specification (which is a partial state specification and not necessarily a state). Any state s, that satisfies the goal specification, i.e., $s \supseteq \mathcal{G}$ is referred to as a goal state. We define the effects of executing an action using a transition function $\mathcal{T}_{\mathcal{M}} : 2^{\mathcal{F}} \times A \rightarrow 2^{\mathcal{F}}$, where

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

where $exec(s,a)$ returns true if $s \supseteq pre_+(a)$ and $s \cap pre_-(a) = \emptyset$. We will overload the notation and use the transition function to also apply to action sequences, where $\mathcal{T}_{\mathcal{M}}(s, \langle a_1, ..., a_k \rangle) = \mathcal{T}_{\mathcal{M}}(...(\mathcal{T}_{\mathcal{M}}(s, a_1), ..., a_k))$. A solution to a planning problem is a plan, which is an action sequence whose execution in $\mathcal{I}$ results in a goal state, i.e., π is a plan if $\mathcal{T}_{\mathcal{M}}(\mathcal{I}, \pi) \supseteq \mathcal{G}$. 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 $\Pi_{\mathcal{M}}$ for the set of all plans.

Design to Reduce Goal State Divergence

Our primary goal here is to identify and avoid scenarios where the goal state that a robot achieves in response to a goal specification differs from the state the human expected. As such, we will introduce a metric to quantify such mismatches. First, we will quantify a mismatch between states by finding their symmetric difference or the elements present in either state but not both. More formally

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.,* $\mathcal{SD}(s^1, s^2) = s^1 \cup s^2 \setminus (s^1 \cap s^2)$

In this paper, we are not interested in measuring the difference between two arbitrary states but rather the goal state

expected by the human and the goal state that the robot will achieve. We will define goal state divergence in terms of plans and their corresponding 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 $\mathcal{M}^2$. Here, 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.:*

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

Here, models of interest correspond to the robot's model of the task $\mathcal{M}^{\mathcal{R}} = \langle \mathcal{D}^{\mathcal{R}}, \mathcal{I}^{\mathcal{R}}, \mathcal{G}^{\mathcal{R}} \rangle$, and the model that captures the human's beliefs about the robot and task $\mathcal{M}^{\mathcal{H}} = \langle \mathcal{D}^{\mathcal{H}}, \mathcal{I}^{\mathcal{H}}, \mathcal{G}^{\mathcal{H}} \rangle$. We assume that the human and robot models share the same fluents $\mathcal{F}$ to simplify the notations. Since the robot is following the goal specification provided by the human, we have $\mathcal{G}^{\mathcal{H}} = \mathcal{G}^{\mathcal{R}}$. However, having the same goal specification doesn't mean that the final goal state expected by the human is the same as the one that will be achieved by the robot, which will depend on the specific plans and models. Goal specification lists a set of fluents that needs to be satisfied for a state to be considered a goal state. Some states might contain undesirable fluents in addition to those listed in the goal specification. 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 $\mathcal{GSD}(\pi^{\mathcal{H}}, \mathcal{M}^{\mathcal{H}}, \pi^{\mathcal{R}}, \mathcal{M}^{\mathcal{R}})$. This captures the divergence between the goal state obtained by executing the human plan in their model and the goal state achieved by the robot plan. In the greenhouse example, $\mathcal{GSD}$ is greater than one since the robot plan might result in a goal state that contains fluents like `heatlamps_wet`, which is absent from the human's expected goal state.

Calculating the above difference requires the system to access both the human model ($\mathcal{M}^{\mathcal{H}}$) and the plan expected by them ($\pi^{\mathcal{H}}$). We will assume access to $\mathcal{M}^{\mathcal{H}}$, along with the robot model. Notably, there are model learning methods we could employ to learn $\mathcal{M}^{\mathcal{H}}$ (cf. (Callanan et al. 2022)). Additionally, for many structured settings where the human model may be known beforehand. In the greenhouse case, if humans have been working with a previous robot model, their beliefs about the robot would be dictated by the capabilities of the previous model. It is worth noting that access to a $\mathcal{M}^{\mathcal{H}}$ doesn't mean you know $\pi^{\mathcal{H}}$. There could be many valid plans in $\mathcal{M}^{\mathcal{H}}$, including multiple optimal ones. Also, in cases where the design is being performed for a set of potential humans, even with similar background knowledge, their expectations about the exact behavior might differ.

Approximating Goal State Divergence

Even if we had access to $\pi^{\mathcal{R}}$, which we do not assume, lack of $\pi^{\mathcal{H}}$ means that we cannot exactly compute $\mathcal{GSD}$. Instead, we will consider useful approximations of $\mathcal{GSD}$. First, we will consider the worst-case approximation or the maximum divergence possible between plans in the two models:

Definition 3. *For two models, $\mathcal{M}^1$, $\mathcal{M}^2$, the maximal goal state divergence ($\mathcal{GD}^{\uparrow}$) is the cardinality of the maximum $\mathcal{GSD}$ possible between all executable plans in $\mathcal{M}^1$ and $\mathcal{M}^2$:*

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

From the definition, we can see that $\mathcal{GD}^{\uparrow}$ will be an upper bound of the true $\mathcal{GSD}$, i.e., $\mathcal{GD}^{\uparrow} \geq |\mathcal{GSD}(\pi^{\mathcal{H}}, \mathcal{M}^{\mathcal{H}}, \pi^{\mathcal{R}}, \mathcal{M}^{\mathcal{R}})|$. As such, one way to reduce $\mathcal{GSD}$ is to reduce $\mathcal{GD}^{\uparrow}$. Especially if we can reduce $\mathcal{GD}^{\uparrow}$ to zero, we are guaranteed that $\mathcal{GSD}$ will be empty. However, $\mathcal{GD}^{\uparrow}$ could be a loose bound, and reducing $\mathcal{GD}^{\uparrow}$ need not reduce $\mathcal{GSD}$. Next, we consider the lower bound on $\mathcal{GSD}$, in which we consider the minimum divergence.

Definition 4. *Given two models, $\mathcal{M}^1$, $\mathcal{M}^2$, the minimal goal state divergence ($\mathcal{GD}^{\downarrow}$) is the cardinality of the minimum $\mathcal{GSD}$ possible between all plans in $\mathcal{M}^1$, and $\mathcal{M}^2$:*

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

With $\mathcal{GD}^{\downarrow}$ in place as a lower bound along with the upper bound $\mathcal{GD}^{\uparrow}$, we can define the design problem.

As discussed, we aim to find a set of model designs (we use design and environment modification/updates interchangeably) to help reduce $\mathcal{GSD}$. Environment design as an offline solution strategy is particularly cost-effective when different humans and possibly different robots repeatedly engage in the same task(s). In the case of the greenhouse, it may be shared by a community, where each human has a robotic assistant. Design allows us to change the greenhouse once instead of updating each robot.

In the greenhouse, the designs include moving objects like the watering pail or hose around or removing them completely. In theory, a design could involve changing any aspect of a planning problem; however, in practice, they are constrained by what a designer could reasonably change. Many existing works focus on design changes that toggle robot action availability. It is also common to associate costs with design to highlight the difficulty of implementing a design. Formally, our design problem is defined as:

Definition 5. *A **human-robot goal-state 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 or model updates. These may include changes to the state space, sensors, 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. It returns the model obtained by incorporating the given modifications.*
- *$\mathbb{U} \rightarrow \mathbb{R}^+$ the cost function for the modifications.*

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 l and upper bound u to fall below a specific threshold.

Definition 6. *An (ℓ, u)-bounded minimal solution to a $\mathcal{DP}$ is a set of modifications [1] ξ^* that minimizes the following constrained optimization problem.*

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

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

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

While the above definitions look at identifying designs for a single task, our framework can be extended to support cases where the robot might be tasked with carrying out one of a set of possible tasks in the same environment. For example, in the greenhouse scenario, the robot may be tasked with watering plants, adding fertilizers, cleaning the floor, etc. We can support such settings by considering how the designs change $\mathcal{GD}^{\downarrow}$ and $\mathcal{GD}^{\uparrow}$ values across the tasks. This means the solutions for such extensions will be almost identical to the ones discussed in the next section.

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

Here, we will employ off-the-shelf cost-optimal planners to calculate $\mathcal{GD}^{\downarrow}$ and $\mathcal{GD}^{\uparrow}$. We will do so by creating a single compiled planning problem that includes actions from both the robot and human models. The compilation will require a plan that includes both a human and a robot plan to the goal, each operating over a different copy of the state fluents. Once the plans are identified, a set of compare actions will be used to compare the final states. To identify the lower bound ($\mathcal{GD}^{\downarrow}$), the planner is penalized when the plans identified result in different goal states (in terms of the values associated with fluents not part of the goal specification). For the upper bound ($\mathcal{GD}^{\uparrow}$), they are penalized for overlap.

More formally, for $\lambda = \langle \mathcal{M}^{\mathcal{R}}, \mathcal{M}^{\mathcal{H}} \rangle$, we create a new compiled model $\mathcal{M}^{\lambda} = \langle \mathcal{D}^{\lambda}, \mathcal{I}^{\lambda}, \mathcal{G}^{\lambda} \rangle$, where $\mathcal{D}^{\lambda} = \langle \mathcal{F}^{\lambda}, \mathcal{A}^{\lambda} \rangle$. 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 corresponding to the robot, and $\mathcal{F}^{\mathcal{H}}$ is a copy of these fluents corresponding to the human's beliefs. We will use these to track how a given plan unfolds according to the human model. We will denote the human copy of a fluent $f^{\mathcal{R}}_i \in \mathcal{F}^{\mathcal{R}}$ as $f^{\mathcal{H}}_i$. $\mathcal{F}^{\theta}$ includes the housekeeping fluents $\mathcal{R}\_can\_act$ and $\mathcal{H}\_can\_act$, which control when a human or robot can perform actions. $\mathcal{F}^{\kappa}$ contains a compare fluent for every fluent in $\mathcal{F}^{\mathcal{R}}$, i.e., $\exists f^{\kappa}_i \in \mathcal{F}^{\kappa}$, for every $f^{\mathcal{R}}_i \in \mathcal{F}^{\mathcal{R}}$. For example, if `plants_watered` was a fluent used in the original greenhouse example, then `plants_watered` will be part of $\mathcal{F}^{\mathcal{R}}$, `human_copy_plants_watered` will be part of $\mathcal{F}^{\mathcal{H}}$, and κ`_plants_watered` will be part of $\mathcal{F}^{\kappa}$. The compare fluents will allow us to track whether each fluent in the resultant goal state from the robot plan and the one expected by the human has been compared. Fig. 2 visualizes the compilation.

$\mathcal{I}^{\lambda}$ is an initial state denoted by $\mathcal{I}^{\lambda} = \mathcal{I}^{\mathcal{R}} \cup \mathcal{I}^{\mathcal{H}} \cup \{\mathcal{H}\_can\_act\}$, 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. For the greenhouse example, $\mathcal{I}^{\mathcal{R}}$ and $\mathcal{I}^{\mathcal{H}}$ are identical, except they are represented using two different sets of fluents. The inclusion of $\{\mathcal{H}\_can\_act\}$, ensures that the plan starts with human actions. $\mathcal{G}^{\lambda}$ is the set of goals shared by the human and robot, denoted by $\mathcal{G}^{\lambda} = \mathcal{G}^{\mathcal{R}} \cup \mathcal{G}^{\mathcal{H}} \cup \mathcal{F}^{\kappa}$. Here, $\mathcal{G}^{\mathcal{R}} \subseteq \mathcal{F}^{\mathcal{R}}$ is the goal specified in the original features, and $\mathcal{G}^{\mathcal{H}} \subseteq \mathcal{F}^{\mathcal{H}}$ is the same goal expressed in the human fluent copy, while $\mathcal{F}^{\kappa}$ are the compare fluents used to determine how similar the human and robot's final goal states are. For the greenhouse, this means achieving `plants_watered` (the specified goal), `human_copy_plants_watered` and one compare fluent for every original fluent.

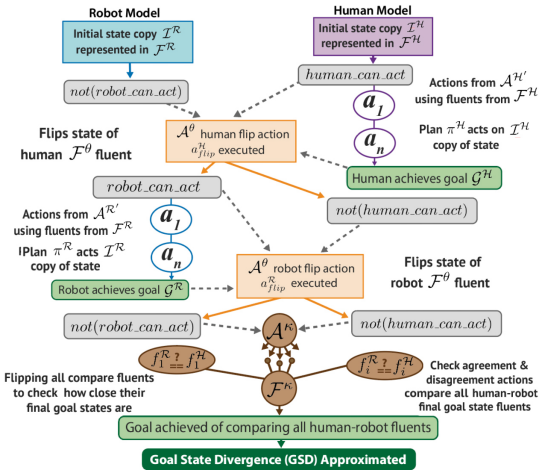

Figure 2: Our $\mathcal{GSD}$ compilation uses two copies of the state operated on by two copies of the actions. Finally, the end states are compared to identify the degree of mismatch. The costs of agreement/disagreement actions can be changed to control whether the upper or lower bound is calculated.

$\mathcal{A}^{\lambda}$ is a set of actions represented by $\mathcal{A}^{\lambda} = \mathcal{A}^{\mathcal{R}'} \cup \mathcal{A}^{\mathcal{H}'} \cup \mathcal{A}^{\theta} \cup \mathcal{A}^{\kappa}$. Here, $\mathcal{A}^{\mathcal{R}'}$ is the action set corresponding to the robot actions $\mathcal{A}^{\mathcal{R}}$. These action definitions are identical to their definitions in $\mathcal{A}^{\mathcal{R}}$, except that for any $a \in \mathcal{A}^{\mathcal{R}'}$, you have $\mathcal{R}\_can\_act \in pre(a)$. Similarly, $\mathcal{A}^{\mathcal{H}'}$ is a copy of these actions corresponding to the human's beliefs of them (i.e., the definitions in $\mathcal{A}^{\mathcal{H}}$) expressed in $\mathcal{F}^{\mathcal{H}}$. Additionally, for any $a \in \mathcal{A}^{\mathcal{H}'}$, you have $\mathcal{H}\_can\_act \in pre(a)$. In the greenhouse example, $\mathcal{A}^{\mathcal{R}'}$ will contain an action `use_hose`, while that action will be missing from $\mathcal{A}^{\mathcal{H}'}$ because the human is not aware the robot can use a hose.

$\mathcal{A}^{\theta}$ are the special flip actions $a^{\mathcal{R}}_{flip}$ and $a^{\mathcal{H}}_{flip}$ 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 the human flip action by:

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

Once the robot has achieved its goals, the action $a^{\mathcal{R}}_{flip}$ is used to disable its ability to perform actions (defined similar to $a^{\mathcal{H}}_{flip}$). Such actions ensure that the planner has identified valid human/robot plans before executing the check actions.

Once the human and robot have both executed their plans, all features from their final goal states are compared. Here, one check fluent action exists for each fluent in $\mathcal{F}^{\mathcal{R}}$. $\mathcal{A}^{\kappa}$ is a set of compare actions that do these comparisons and generate the check fluents. $\mathcal{A}^{\kappa}$ is denoted as $\mathcal{A}^{\kappa} = A^{\kappa}_{f_1} \cup A^{\kappa}_{f_2} \cup A^{\kappa}_{f_3} ... A^{\kappa}_{f_{|\mathcal{F}|}}$, such that the set $\mathcal{A}^{\kappa}_{f_i} = \{a^1_{f_i}, a^2_{f_i}, a^3_{f_i}, a^4_{f_i}\}$ exists for each $f^{\mathcal{R}}_i \in \mathcal{F}^{\mathcal{R}}$. 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 execute if the human's belief about the fluent value matches the robot's, and the disagreement copies only when 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. The definition for $a^1_{f_i}$ (checks if the robot copy fluent is true and the human copy is false) and $a^3_{f_i}$ (checks if both copies are true) are given as follows:

- $pre_+(a^1_{f_i}) = \{f^{\mathcal{R}}_i\}$, $pre_-(a^1_{f_i}) = \{f^{\mathcal{H}}_i\} \cup \{\mathcal{R}\_can\_act\} \cup \{\mathcal{H}\_can\_act\} \cup \{f^{\kappa}_i\}$, $add(a^1_{f_i}) = \{f^{\kappa}_i\}$, $del(a^1_{f_i}) = \emptyset$, and $c(a^1_{f_i}) = \mathcal{P}_1$
- $pre_+(a^3_{f_i}) = \{f^{\mathcal{R}}_i, f^{\mathcal{H}}_i\}, pre_-(a^3_{f_i}) = \{\mathcal{R}\_can\_act\} \cup \{\mathcal{H}\_can\_act\} \cup \{f^{\kappa}_i\}$, $add(a^3_{f_i}) = \{f^{\kappa}_i\}$, $del(a^3_{f_i}) = \emptyset$, and $c(a^3_{f_i}) = \mathcal{P}_2$

The actions $a^2_{f_i}$ and $a^4_{f_i}$ are defined in a similar fashion. $a^2_{f_i}$ will check if the human copy is true and the robot copy is false, while $a^4_{f_i}$ checks if both values are false. 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. Note that the addition of $\mathcal{F}^\kappa$ to the goal means that one of the possible actions from $A^\kappa_{f_i}$ has to be part of the plan for every $f^{\mathcal{R}}_i \in \mathcal{F}^{\mathcal{R}}$. This is always possible given that the individual actions in $A^\kappa_{f_i}$, capture all possible combinations in which fluents could occur in the human and robot part of the state.

Finally, for the cost function, we will set all robot/human actions to a unit cost and the $\mathcal{A}^\theta$ actions to zero. We will use the compilation to generate the lower bound ($\mathcal{GD}^\downarrow$) by giving high costs to disagreement actions and the upper bound ($\mathcal{GD}^\uparrow$) by assigning high costs to agreement actions.

Proposition 1. *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. This is because check agreement actions are dominating the plan's cost. Here, the cost of a single agreement action is higher than the combined cost of the longest possible plan in $\mathcal{M}^{\mathcal{H}}$ or $\mathcal{M}^{\mathcal{R}}$ (i.e., one that passes through every state). Thus, we force the planner to select plans with the lowest possible degree of agreement. □

By inverting costs, we can also calculate $\mathcal{GD}^\downarrow$:

Proposition 2. *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.

Identifying Minimal Designs for HRGAD

With the methods in place for calculating our $\mathcal{GD}^\uparrow$ and $\mathcal{GD}^\downarrow$ bounds, we now look to identify designs that will meet the criteria for a Human-Robot Goal-State Alignment design (*HRGAD*) problem. We consider a specific instantiation of Definition 6, where the bounds on $\mathcal{GD}^\downarrow$, i.e., ℓ is set to zero, design set $\mathbb{U}$ is limited to initial state changes and each design has a unit cost. To find this set, an algorithm with an outer and an inner loop is employed. In the outer loop, a design budget τ is set to one, which iteratively increases. Given the current design budget, the inner loop tries to identify a design where the $\mathcal{GD}^\downarrow$ is zero, and the $\mathcal{GD}^\uparrow$ is within the specified limit u. In this section, we detail this inner loop and extend our previously compiled model to automatically identify designs that result in models where $\mathcal{GD}^\downarrow$ is zero.

Inner Loop for Identifying Designs

In the inner loop, the design phase begins, where the $\mathcal{GD}^\downarrow$ compilation is modified to automatically identify the designs. In the original compilation, the objective was only to approximate $\mathcal{GSD}$, but we will now extend this compilation by adding a design phase before it. Using this updated compilation, a set of designs is found that results in $\mathcal{GD}^\downarrow$ of zero. This set of identified designs is then inputted into the $\mathcal{GD}^\uparrow$ part of the inner loop and checked to see if they fall within the required u-upper-bound. If they do, the current set of designs is returned as the final set. If they don't, the compilation is updated to disallow these design modifications, and the updated $\mathcal{GD}^\downarrow$ tries to find another design. If no more designs that result in zero $\mathcal{GD}^\downarrow$ are found, then the budget τ is increased and the process repeats.

Extended Compiled Model To find the set of all design modifications that fit the budget and satisfy the $\mathcal{GD}^\downarrow = 0$ requirement, the previously compiled model $\mathcal{M}^\lambda$, is extended into $\mathcal{M}^\lambda_{\mathbb{U}} = \langle \mathcal{D}^\lambda_{\mathbb{U}}, \mathcal{I}^\lambda_{\mathbb{U}}, \mathcal{G}^\lambda_{\mathbb{U}} \cup \{unseen\_design\} \rangle$. In $\mathcal{M}^\lambda_{\mathbb{U}}$, the planner will select the design per the model update set $\mathbb{U}$ through a set of design actions. The planner's new goal is then to select a set of design actions and a pair of human-robot plans whose resulting goal states match exactly. We ensure this latter condition by removing all disagreement actions from the compare action set. This means the $\mathcal{F}^\kappa$ part of the goal can only be satisfied if all fluent values in the resultant goal states match, i.e., $\mathcal{GD}^\downarrow = 0$. This model will also allow us to skip previously seen designs by tracking them and having a conditional effect that only allows the plan to proceed if the designs considered are not in this previously seen set. It will also use budget fluents in the design action to ensure only budgets of a specified size are considered.

Now, the new model's domain $\mathcal{D}^\lambda_{\mathbb{U}}$ 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. The set $\mathcal{F}^\lambda_{\mathbb{U}}$ is now characterized by $\mathcal{F}^\lambda_{\mathbb{U}} = \mathcal{F}^\lambda \cup \{design\_allowed\} \cup \{unseen\_design\} \cup \mathcal{F}^\tau \cup \mathcal{F}^{\tau+}$. $\mathcal{F}^\lambda$ 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 the design actions can be executed. The $\{unseen\_design\}$ fluent is used to indicate that a given design has not been seen or

used before and, as such, could be a candidate for reducing $\mathcal{GSD}$. This fluent begins as true in the initial state and may be deleted if the plan uses a design seen before. $\mathcal{F}^{\tau}$ and $\mathcal{F}^{\tau+}$, are together used to ensure that the planner identifies a design set of size τ. Here, $\mathcal{F}^{\tau}$ and $\mathcal{F}^{\tau+}$ contains τ fluent and will be used to enforce the design budget. Each design action will require fluents from $\mathcal{F}^{\tau}$ as a precondition, which the action will delete, and then we will add a fluent from $\mathcal{F}^{\tau+}$. In the goal, we can then force that τ design actions have been performed by simply adding $\mathcal{F}^{\tau+}$ to the goal.

$\mathcal{A}^{\lambda}_{\mathbb{U}} = ((\mathcal{A}^{\lambda} \cup \mathcal{A}^{\mathbb{U}}) \setminus \mathcal{A}^{\kappa-}) \cup \{design\_completed\}$ is the updated set of actions. $\mathcal{A}^{\lambda}$ 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 $\mathcal{A}^{\kappa}$). $|\mathcal{A}^{\mathbb{U}}| = \tau \times |\mathbb{U}|$ is the available set of design actions that must be executed for a given modification to be made. Recall, we remove the subset of check disagreement actions, $\mathcal{A}^{\kappa-}$, to only allow for selecting human-robot plan pairs whose final goal states match exactly, which only requires check agreement compare actions. Each design action $a \in \mathcal{A}^{\mathbb{U}}$ corresponds to a specific model update, i.e., a change in the initial state and a specific step t_i in the total allowed design size of τ. Additionally, the design actions are only allowed during the design phase, which is denoted by the fluent $design\_allowed$ being true. Thus, for an individual design action corresponding to time step t_i, the positive precondition will be $\{design\_allowed,\ f_i\}$, where $f_i \in \mathcal{F}^{\tau}$. Once executed, the design action will either add or remove a fluent from the initial state according to the corresponding model update (this will be captured by a corresponding add or delete effect). Once a design step for t_i is executed, no other design action should be executed for that time step. As discussed, we will allow this by making sure the action deletes the fluent $f_i \in \mathcal{F}^{\tau}$. The action will add $f_i^{+} \in \mathcal{F}^{\tau+}$, which will be used in the goal specification to track that a design of a specific size was created.

The $\{design\_completed\}$ action will end the design phase after all modifications have been made by deleting the $\{design\_allowed\}$ fluent. To disallow previously seen designs, we will also introduce conditional effects into the $\{design\_completed\}$ action, where the condition corresponds to the design fluents of a previously identified design, and the effect is to delete the $\{unseen\_design\}$ fluent. This action will also add the $\{\mathcal{H}\_can\_act\}$ fluent which will resume planning as described in the previous section.

The initial state for the new problem is given as: $\mathcal{I}^{\lambda}_{\mathbb{U}} = (\mathcal{I}^{\lambda} \setminus \{\mathcal{H}\_can\_act\}) \cup \{unseen\_design, design\_allowed\} \cup \mathcal{F}^{\tau}$. $\mathcal{I}^{\lambda}$ 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 $\{\mathcal{H}\_can\_act\}$ fluent removed. This is because plans for $\mathcal{M}^{\lambda}_{\mathbb{U}}$ initially start in the design phase, where modifications are being made to reduce $\mathcal{GSD}$, and neither the human nor the robot can act. We do this by adding $\{design\_allowed\}$ to the initial state. The addition of $\{unseen\_design\}$ ensures that a design is not seen before if none of the conditional effects in the $\{design\_completed\}$ action removes it. Lastly, we add $\mathcal{F}^{\tau}$ to initial state fluents to track the allowed design size. $\mathcal{G}^{\lambda}_{\mathbb{U}} = \mathcal{G}^{\lambda} \cup \mathcal{F}^{\tau+} \cup \{unseen\_design\}$ is the updated goal. $\mathcal{G}^{\lambda}$ 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 $\mathcal{F}^{\kappa}$), with $\mathcal{F}^{\tau+}$ now added, indicating that τ designs must be applied. All actions in this new model have a unit cost.

Extracting and Using the Designs. The above compilation will only return a solution if it identifies design modifications that result in $\mathcal{GD}^{\downarrow} = 0$. The designs are then extracted and applied in the initial state, and the updated model is passed to the $\mathcal{GD}^{\uparrow}$ compilation. If the bound for $\mathcal{GD}^{\uparrow}$ is within the upper-bound u the algorithm stops and returns the identified set of design modifications. We resolve the compiled planning problem if no modification set exists that meets the upper-bound requirement. To ensure that the compilation doesn't return a previously identified compilation, we use a set of conditional effects within the $design\_completed$ action which checks if the currently identified modification set corresponds to a previously seen modification. If it does, then the conditional effect makes the $\{unseen\_design\}$ fluent false. Since $\{unseen\_design\}$ is part of the goal, the compilation cannot achieve the goal using the current modifications, thus forcing the planner to identify alternate designs. An unsolvable compilation implies that there are no more designs of budget τ left that can make $\mathcal{GD}^{\uparrow} = 0$, and the budget needs to be increased.

Evaluation

Our evaluation objective was to provide a computational characterization of the methods for computing $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$ and identifying design modifications. Since our methods are guaranteed to find the minimal design that meets the required $\mathcal{GD}^{\downarrow}$and $\mathcal{GD}^{\uparrow}$limit if one exists, our primary metric of evaluation will be the time taken by each method. The supplementary file also lists some additional experiments related to the size of the problems and designs.

Dataset We considered five International Planning Competition (IPC) domains (ICAPS 2016a,b), and used five problem instances from each domain to create goal state divergence problems. The original instances were used in the robot model, while the human models were formed by deleting five random initial state fluents from the initial state of the original instance. We repeated this five times to create five unique design problems from the original problem instance.Each row in Table 1 reports values averaged across the five variations for each instance. Our setup allows us to guarantee that there always exists a design that results in $\mathcal{GD}^{\downarrow} = 0$. For consistency, we considered a $\mathcal{GD}^{\uparrow}$ 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. By forcing both approximations to be zero, our evaluations are looking for a design that avoids any misspecification. Thus allowing us to perform cross-domain comparisons while keeping the $\mathcal{GD}^{\uparrow}$ constant. We were able to guarantee the existence of designs that can $\mathcal{GD}^{\uparrow} = 0$ limit, by updating the goal specification.

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 all methods compared to each baseline in seconds. The first three columns show the time taken by our method, a variation of it without enforced ordering, and a baseline that iterates over possible designs. The remaining columns show the time to compute the lower $\mathcal{GSD}$ bound, lower bound with design, and upper bound.

Setup We implemented the compilations for individually $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$, and the updated $\mathcal{GD}^{\downarrow}$ compilation that also identifies the design. We also implement a simple breadth-first search over the design space as the baseline. The search ends whenever it finds a design with zero upper and lower bounds. For design, our primary points of comparison will be our proposed algorithm (referred to as `Main`), and we will refer to our breadth-first search baseline as (`Naive`). We will also consider a variation of the `Main` that considers a flattened compilation (`Main-fl`), where we ignore the ordering between the human and robot actions. 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 against which to compare this work. 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, 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 shorter time than the baseline**, and were mostly equivalent. As such, we see that enforcing the ordering doesn't seem to make much of a difference when compared to the flattened compilation. Compared to the naive baseline, we see our method takes significantly less time. This shows the advantage of using planners to identify possible designs. We see the most noticeable benefit in the Depot domain. The addition of design into $\mathcal{GD}^{\downarrow}$ compilation also seems to add minimal overhead. Finally, the $\mathcal{GD}^{\uparrow}$ times are, in general, higher than $\mathcal{GD}^{\downarrow}$ times. However, this is expected in this setting as calculating $\mathcal{GD}^{\uparrow}$ here involves testing for unsolvability.

Conclusion

The paper focuses on providing a framework to understand and study environment design for generating agent behavior that aligns with human goal state expectations. The specific design problem we focus on is one among several we could study in this space. Next, we hope to look at more complex plannign settings. 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.

Acknowledgements

This work was partially funded by NSF grant 2303019.

References

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Mané, D. 2016. Concrete Problems in AI Safety. *CoRR*, abs/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. *CoRR*, abs/2206.06530.

Gabriel, I.; and Ghazavi, V. 2023. The Challenge of Value Alignment: From Fairer Algorithms to AI Safety. In *Oxford Handbook of Digital Ethics*. Oxford University Press. ISBN 9780198857815.

Hadfield-Menell, D.; Russell, S.; Abbeel, P.; and Dragan, A. D. 2016. Cooperative Inverse Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 3909–3917. Curran Associates, Inc.

Hanni, A.; Boateng, A.; and Zhang, Y. 2024. Safe explicable planning. In *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling*, ICAPS '24. AAAI Press. ISBN 1-57735-889-9.

Hanni, A.; Montaño, J.; and Zhang, Y. 2025. Safe Explicable Policy Search. arXiv:2503.07848.

ICAPS. 2016a. 2nd International Planning Competition, 2000.

ICAPS. 2016b. 3rd International Planning Competition, 2002. Github.

Ji, J.; Qiu, T.; Chen, B.; Zhou, J.; Zhang, B.; Hong, D.; Lou, H.; Wang, K.; Duan, Y.; He, Z.; Vierling, L.; Zhang, Z.; Zeng, F.; Dai, J.; Pan, X.; Xu, H.; O'Gara, A.; Ng, K.; Tse, B.; Fu, J.; McAleer, S.; Wang, Y.; Yang, M.; Liu, Y.; Wang, Y.; Zhu, S.-C.; Guo, Y.; Yang, Y.; and Gao, W. 2025. AI Alignment: A Contemporary Survey. *ACM Comput. Surv.*

Keren, S.; Gal, A.; and Karpas, E. 2014. Goal Recognition Design. In *Proceedings of ICAPS*, 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 Maximized Utility. *Proceedings of AAAI*, 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 IJCAI*, 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 *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. 2024. Goal Alignment: Re-analyzing Value Alignment Problems Using Human-Aware AI. In *Proceedings of AAAI*, 10110–10118. AAAI Press.

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).

Monette, N.; Letcher, A.; Beukman, M.; Jackson, M. T.; Rutherford, A.; Goldie, A. D.; and Foerster, J. 2025. An Optimisation Framework for Unsupervised Environment Design. *arXiv.org*.

Pozanco, A.; Fraga Pereira, R.; and Borrajo, D. 2024. Generalising Planning Environment Redesign. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18): 20230–20237.

Reddy, S.; Dragan, A.; and Levine, S. 2018. Where do you think you're going?: Inferring beliefs about dynamics from behavior. *Advances in Neural Information Processing Systems*.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *JAIR*, 127–177.

Saisubramanian, S.; and Zilberstein, S. 2021. Mitigating Negative Side Effects via Environment Shaping. In *AAMAS*, AAMAS '21, 1640–1642. Richland, SC: IFAAMAS. 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.

Sorensen, T.; Moore, J.; Fisher, J.; Gordon, M.; Mireshghallah, N.; Rytting, C. M.; Ye, A.; Jiang, L.; Lu, X.; Dziri, N.; Althoff, T.; and Choi, Y. 2024. A Roadmap to Pluralistic Alignment. arXiv:2402.05070.

Sreedharan, S. 2023. *Human-aware AI* - A foundational framework for human-AI interaction. *AI Mag.*, 44(4): 460–466.

Sreedharan, S. 2025. Using Human-Aware AI as a Framework to Achieve Intent Alignment. In *2025 IEEE Conference on Artificial Intelligence (CAI)*, 1217–1220.

Sreedharan, S.; Hernandez, A. O.; Mishra, A. P.; and Kambhampati, S. 2019. Model-Free Model Reconciliation.

In *Proceedings of IJCAI*, 587–594. AAAI Press. ISBN 9780999241141.

Steinmetz, M.; Hoffmann, J.; Kovtunova, A.; and Borgwardt, S. 2022. Classical Planning with Avoid Conditions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9): 9944–9952.

Teoh, J.; Li, W.; and Varakantham, P. 2025. Improving Environment Novelty Quantification for Effective Unsupervised Environment Design. *Neural Information Processing Systems*.

Wang, R.; Lehman, J.; Clune, J.; and Stanley, K. O. 2019. Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions. *arXiv: Artificial Intelligence*.

Wang, R.; Lehman, J.; Rawal, A.; Zhi, J.; Li, Y.; Clune, J.; and Stanley, K. O. 2020. Enhanced POET: Open-Ended Reinforcement Learning through Unbounded Invention of Learning Challenges and their Solutions. *International Conference on Machine Learning*.

Weld, D.; and Etzioni, O. 1994. The First Law of Robotics (a Call to Arms). In *Proceedings of AAAI*, 1042–1047. AAAI Press.

Zhang, H.; and Parkes, D. 2008. Value-Based Policy Teaching with Active Indirect Elicitation. In *AAAI*, 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 IJCAI*, 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 *ICRA*, 1313–1320.