Main Supplementary file for paper ID 27920

Kelsey Sikes, Sarah Keren, Sarath Sreedharan

1 Greenhouse Disaster Model Compilation

Model compilation is a technique for simplifying a planning problem so it can be used more easily by a planner. In Section 4 of the main paper, we illustrated how environment design might be used to influence a robot's decision-making in a greenhouse setting. Here we map the Goal State Divergence ($\mathcal{GSD}$) model compilation to that example. We begin with the compiled model $\mathcal{M}^\lambda = \langle \mathcal{D}^\lambda, \mathcal{I}^\lambda, \mathcal{G}^\lambda \rangle$, where $\mathcal{D}^\lambda$ is defined by the tuple $\mathcal{D}^\lambda = \langle \mathcal{F}^\lambda, \mathcal{A}^\lambda \rangle$ which contained fluents and actions, $\mathcal{I}^\lambda$ is some initial state, and $\mathcal{G}^\lambda$ a goal specification.

1.1 Compiled Fluents

$\mathcal{F}^\lambda$ is a set of compiled fluents, shown by $\mathcal{F}^\lambda = \mathcal{F}^\mathcal{R} \cup \mathcal{F}^\mathcal{H} \cup \mathcal{F}^\theta \cup \mathcal{F}^\kappa$. Here, $\mathcal{F}^\mathcal{R}$ corresponds to the original set of robot fluents, individually denoted by $f_i^\mathcal{R}$ (i.e. $f_i^\mathcal{R} \in \mathcal{F}^\mathcal{R}$). These are used to keep track of how the robot's plan and final goal state unfolds. To illustrate this, imagine there is a greenhouse plant in the robot's model called `plant1` that needs to be watered. Using the fluent `(robot-watered ?plant)`, we could check if this specific plant was indeed watered at the end of the robot's plan (i.e. if `(robot-watered plant1)` is true). Here, `(robot-watered plant1)` would be representative of an individual $f_i^\mathcal{R}$ fluent in $\mathcal{F}^\mathcal{R}$.

Similarly, $\mathcal{F}^\mathcal{H}$ corresponds to the human's set of belief fluents about the robot, individually denoted by $f_i^\mathcal{H}$. These fluents are an exact copy of the robot's fluents in $\mathcal{F}^\mathcal{R}$, meaning for every $f_i^\mathcal{R} \in \mathcal{F}^\mathcal{R}$, a $f_i^\mathcal{H}$ copy fluent exists. These are used to keep track of how the human believes the robot's plan and final goal state will unfold, relative to what they believe its current state and capabilities to be. Like before, imagine there is a plant in the human's greenhouse that needs to be watered. This plant is identical to the one in the robot's greenhouse and is also called `plant1`. Using the fluent `(human-watered ?plant)`, we could check if at the end of the human's plan, this specific plant was watered too (i.e. if `(human-watered plant1)` is true). Here, `(human-watered plant1)` would be representative of an individual $f_i^\mathcal{H} \in \mathcal{F}^\mathcal{H}$.

$\mathcal{F}^\theta$ represents the special housekeeping fluents $\{human\_can\_act\}$ and $\{robot\_can\_act\}$, which control when the human or robot can perform actions. Though not a requirement, this allows an ordering to be enforced between the human and robot's

actions for a more efficient compilation, while also preventing the human and robot from performing additional actions once their final goal states have been achieved. This ensures these states remain unchanged prior to their $\mathcal{GSD}$ being calculated, guaranteeing that the estimation returned by the planner is correct.

$\mathcal{F}^{\kappa}$ corresponds to compare fluents, individually denoted by f_i^{κ} and for which one exists for every $f_i^{\mathcal{R}}$ (i.e. for every $f_i^{\mathcal{R}} \in \mathcal{F}^{\mathcal{R}}, \exists f_i^{\kappa} \in \mathcal{F}^{\kappa}$). We use these to track whether corresponding pairs of fluents from the human and robot's final goal states have been compared. This process sits at the heart of the $\mathcal{GSD}$ metric, which we approximate based on whether the final state of these fluents match or not. Here, `(robot-watered plant1)` and `(human-watered plant1)` would be an example of a corresponding pair of fluents. Though, in our greenhouse setting, there would likely be more, such as `(robot-watered plant2)` and `(human-watered plant2)`. Using the compare fluent, `(water-checked)`, we could track if the final states of all fluent pairs like this have been compared. If so, `(water-checked)` would then become true.

1.2 Compiled Initial State

The initial state in our compiled model is defined by $\mathcal{I}^{\lambda} = \mathcal{I}^{\mathcal{R}} \cup \mathcal{I}^{\mathcal{H}} \cup \{human\_can\_act\}$. Here, $\mathcal{I}^{\mathcal{R}}$ represents the robot's initial state. This is the true version of the world the robot begins operating in at the start of its planning process. In the greenhouse, this may include where certain plants are located, what tools are available to water them with, and what the interior layout of the greenhouse is like. $\mathcal{I}^{\mathcal{H}}$ is then the human's beliefs about the robot's initial state, which is the starting conditions they think the robot is beginning its planning process with. These beliefs may not match true reality, in which case plants in the human's version of the greenhouse may be in different positions, certain tools missing, and its layout completely changed. Here, the special housekeeping fluent $\{human\_can\_act\}$ is also included as part of the initial state. This means the human begins in a state able to perform actions, whereas the robot does not. In the evaluation section of the main paper, this constraint was discussed in more detail. However, if the $\{robot\_can\_act\}$ fluent was added to the initial state, this enforced ordering would be removed. Then the human and robot could both execute actions at the same time.

1.3 Compiled Goals

$\mathcal{G}^{\lambda}$ represents the human and robot's shared goals, defined by $\mathcal{G}^{\lambda} = \mathcal{G}^{\mathcal{R}} \cup \mathcal{G}^{\mathcal{H}} \cup \mathcal{F}^{\kappa}$. $\mathcal{G}^{\mathcal{R}}$ and $\mathcal{G}^{\mathcal{H}}$ correspond to the robot and human's goal specification, each expressed using a subset of their original or copy fluents (i.e. $\mathcal{G}^{\mathcal{R}} \subseteq \mathcal{F}^{\mathcal{R}}$, $\mathcal{G}^{\mathcal{H}} \subseteq \mathcal{F}^{\mathcal{H}}$). This is because while many fluents may describe a state, far fewer are needed to describe a goal. As mentioned earlier, because the objective of this compilation is to compare the human and robot's final goal states, these goal specifications are the same (i.e. $\mathcal{G}^{\mathcal{H}} = \mathcal{G}^{\mathcal{R}}$). Meaning, the human and robot are both trying to achieve the same goal, like watering all plants inside the greenhouse. Also included in this set, to compare these states once all goals have

been achieved, is the original check fluents, $\mathcal{F}^{\kappa}$. This means to calculate $\mathcal{GSD}$, the human and robot must have both achieved their goals and had all of their final goal state fluent pairs like `(robot-watered plant1)` and `(human-watered plant1)` compared, to ensure all $\mathcal{F}^{\kappa}$ fluents like `(water-checked)` became true.

1.4 Compiled Actions

$\mathcal{A}^{\lambda}$ are the actions the robot can perform, as well as those the human believes it can perform, denoted by $\mathcal{A}^{\lambda} = \mathcal{A}^{\mathcal{R}'} \cup \mathcal{A}^{\mathcal{H}'} \cup \mathcal{A}^{\theta} \cup \mathcal{A}^{\kappa}$. Here, $\mathcal{A}^{\mathcal{R}'}$ represents the set of actions executable in the robot's model, which all contain the precondition $\{robot\_can\_act\}$, dictating when they can be performed. These actions contain fluents from $\mathcal{F}^{\mathcal{R}}$.

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

$\mathcal{A}^{\theta}$ are special flip actions that enable or disable the robot or human's ability to perform actions by changing the state of the $\mathcal{F}^{\theta}$ fluents (i.e. changing $\{robot\_can\_act\}$ or $\{human\_can\_act\}$ to true or false). In Section 5.1 of the main paper, we show an example flip action. These actions are denoted by $a^{\mathcal{H}}_{flip}$ and $a^{\mathcal{R}}_{flip}$, and important because they guarantee that a valid plan has been found in the human and robot's models before any final goal state fluents have been compared, and prevent the human and robot from performing any additional actions once their final goal states have been achieved, ensuring the planner can correctly estimate their $\mathcal{GSD}$. In this work, we assume all actions in $\mathcal{A}^{\theta}$ have a unit cost of one.

$\mathcal{A}^{\kappa}$ corresponds to a set of compare actions responsible for comparing all human-robot final goal state fluents, denoted by $\mathcal{A}^{\kappa} = A^{\kappa}_{f_1} \cup A^{\kappa}_{f_2} \cup A^{\kappa}_{f_3} ... A^{\kappa}_{f_{|\mathcal{F}|}}$. Here, for every $f^{\mathcal{R}}_i \in \mathcal{F}^{\mathcal{R}}$, a set of compare actions exists such that $A^{\kappa}_{f_i} = \{a^1_{f_i}, a^2_{f_i}, a^3_{f_i}, a^4_{f_i}\}$. These actions contain $f^{\mathcal{R}}_i$ and $f^{\mathcal{H}}_i$ fluents that are being compared (i.e. `(human-watered plant1)` and `(robot-watered plant1)`), as well as f^{κ}_i fluents for checking if this comparison has been made (i.e. `(water-checked)`). While corresponding fluents may describe the same attributes of a domain, when compared, their respective states may be the same or completely different. We list these variations in Table 1 alongside examples relevant to our greenhouse setting.

To account for these differences, we use unique check agreement and check disagreement compare actions, to analyze whether a pair of corresponding fluents share the same state. These actions only execute when certain preconditions

Scenario	Robot Fluent	Human Fluent	Greenhouse Example
1	True	False	The robot watered plant1 but the human didn't water plant1.
2	False	True	The robot didn't water plant1 but the human did water plant1.
3	True	True	The robot and human both watered plant1.
4	False	False	Neither the robot nor the human watered plant1.

Table 1: Here we show all fluent state combinations, each with an example taken from our greenhouse setting.

are met and contain compare fluents for verifying that these comparisons have taken place. Upon execution, each unique check agreement and check disagreement compare action is assigned a cost, which we represent using the symbols $\mathcal{P}_1$ and $\mathcal{P}_2$. Based on which $\mathcal{GSD}$ bound we are trying to find, these costs are modulated to encourage the planner to select plans with either more agreement or more disagreement between the human and robot's final goal states.

We begin with the check disagreement compare actions, which we denote by $\mathcal{A}^{\kappa-}$. These correlate to the first and second greenhouse scenarios in Table 1. These actions only execute when the human and robot are both in states unable to act, and share corresponding fluents with opposite states, never compared before (i.e. $f_i^{\mathcal{R}}$ must be true when $f_i^{\mathcal{H}}$ is false, or $f_i^{\mathcal{H}}$ must be true when $f_i^{\mathcal{R}}$ is false). When trying to find the $\mathcal{GD}^{\uparrow}$, we assign zero cost to these actions to encourage the planner to select human-robot plans with high levels of disagreement. Conversely, when trying to find the $\mathcal{GD}^{\downarrow}$, we assign a steep cost to these actions to encourage the planner to select human-robot plans with low levels of disagreement. We show these check disagreement compare actions by:

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

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

Next is the check agreement compare actions, which we denote by $\mathcal{A}^{\kappa+}$. These actions correlate to the third and fourth greenhouse scenarios in Table 1. These actions only execute when the human and robot are both in states unable to act, and share corresponding fluents with matching states, never compared before (i.e. $f_i^{\mathcal{R}}$ and $f_i^{\mathcal{H}}$ must both be true, or $f_i^{\mathcal{R}}$ and $f_i^{\mathcal{H}}$ must both be

false). When trying to find the $\mathcal{GD}^{\uparrow}$, we assign a steep cost to these actions to encourage the planner to select human-robot plans with low levels of agreement. Conversely, when trying to find the $\mathcal{GD}^{\downarrow}$, we assign zero cost to these actions to encourage the planner to select human-robot plans with high levels of agreement. We show these check agreement compare actions by:

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

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

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

2 Algorithm

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.

2.1 Theoretical Properties

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

The proof of the above proposition directly follows from the soundness of the individual compilation leveraged in the algorithm. The soundness of the basic compilations is established in Propositions One and Two in the main paper. The only additional extension considered is the fact that the $\mathcal{GD}^{\downarrow}$ compilation considered is one that also identifies the design. Here, the compilation corresponds to two components: the preliminary set of actions that identifies a set of designs and the part that checks the resultant problems $\mathcal{GD}^{\downarrow}$. Now, the former

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

component is similar to previous works that have looked at planning compilations that make problems solvable (cf. Göbelbecker *et al.* [2010]). In this case, the second part will only solve a solvable problem if $\mathcal{GD}^{\downarrow}$ is zero. As such, the overall algorithm will only identify a valid design.

Supplementary Proposition 2. Algorithm 1 is a complete and optimal procedure, so far that it will always find a valid design ξ that meets the criteria for the minimal solution provided by Definition 6 for $\ell = 0$ and k, provided one exists.

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

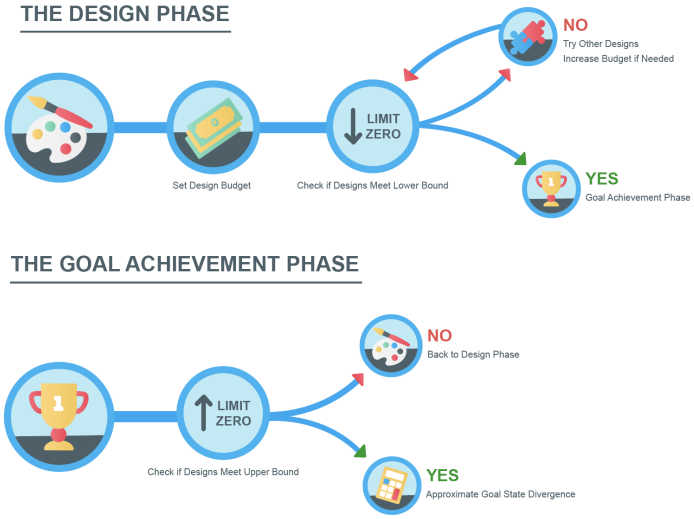


Figure 1: A visualization of the overall algorithmic process.

3 Exposition on the Compilations

3.1 Supporting Plan Subsets

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

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

Where β is a parameter called the rationality parameter. Now we can only consider plans that are above some probability threshold ϵ. For any such probability threshold, there will be a corresponding cost value C_ϵ, such that for any plan π, where $c(\pi) > C_\epsilon$, you will have $P(\pi) < \epsilon$. We will denote the corresponding upper and lower bounds of $\mathcal{GSD}$, where the plan spaces are bounded as described above as $\mathcal{GD}^{\uparrow}_{\langle \epsilon, * \rangle}$ and $\mathcal{GD}^{\downarrow}_{\langle \epsilon, * \rangle}$.

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

3.1.1 Calculating these new bounds

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

Supplementary Proposition 3. For a given compiled model $\mathcal{M}^\lambda$, let us adopt the following cost function

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

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

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

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

3.2 Updated Compilation

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

As with the previous formulation, each design now corresponds to a specific model component. Though now the component can be any part of the model other than the goals. Changes to the initial state are treated as before by actions that directly update the initial state before the *design_completed* action.

For the other model changes, we will again introduce a set of design actions $\mathcal{A}^{\mathcal{D}'}$, that will possibly add a fluent each from the set $\mathcal{D}'_{\mathbb{U}}$. As with previous design actions, we constrain the applicability of these actions to only before the *design_completed* action. Now for each model component that will be influenced by these designs, we will introduce a new model component in the model. These new model components will be conditioned on the new design fluents. For example, let's say a precondition p is added to action a by a design corresponding to fluent $d_i \in \mathcal{A}^{\mathcal{D}'}$, then the human and robot copies of the action $a^{\mathcal{H}}$ and $a^{\mathcal{R}}$ will now contain a precondition $\neg d_i \vee p^{\mathcal{H}}$ and $\neg d_i \vee p^{\mathcal{R}}$ respectively. If the design results in a precondition being removed, the precondition becomes $d_i \vee p^{\mathcal{H}}$ in the human model (similar precondition will be introduced in the robot model). Similarly, for effects, we can introduce conditional effects, where it is conditioned on design fluents from $\mathcal{D}'_{\mathbb{U}}$. Now if we keep the rest of the compilation the same, we can support designs that update any part of the model.

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

4 Additional Experiments

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

4.1 Effect of Problem Size on the Time Taken

First, we will start by looking at the sizes of the problems considered in our evaluation. In particular, we will focus on the number of grounded predicates since the planner we consider operates over grounded planning problems and since the count gives a direct size of the state space. This number was measured after a reachability analysis. Table 2 presents these numbers. Figures 2,3, and 4 provide an overview of the time taken as plotted against the size of the problem. In addition to the total average time taken versus the time taken for individual $\mathcal{GD}^{\uparrow}$ and $\mathcal{GD}^{\downarrow}$ calculations. We see the effect of the size more clearly in the latter calculation time.

We see that apart from logistics, there is some variation in the number of grounded predicates, with the most variation occurring in Zenotravel.

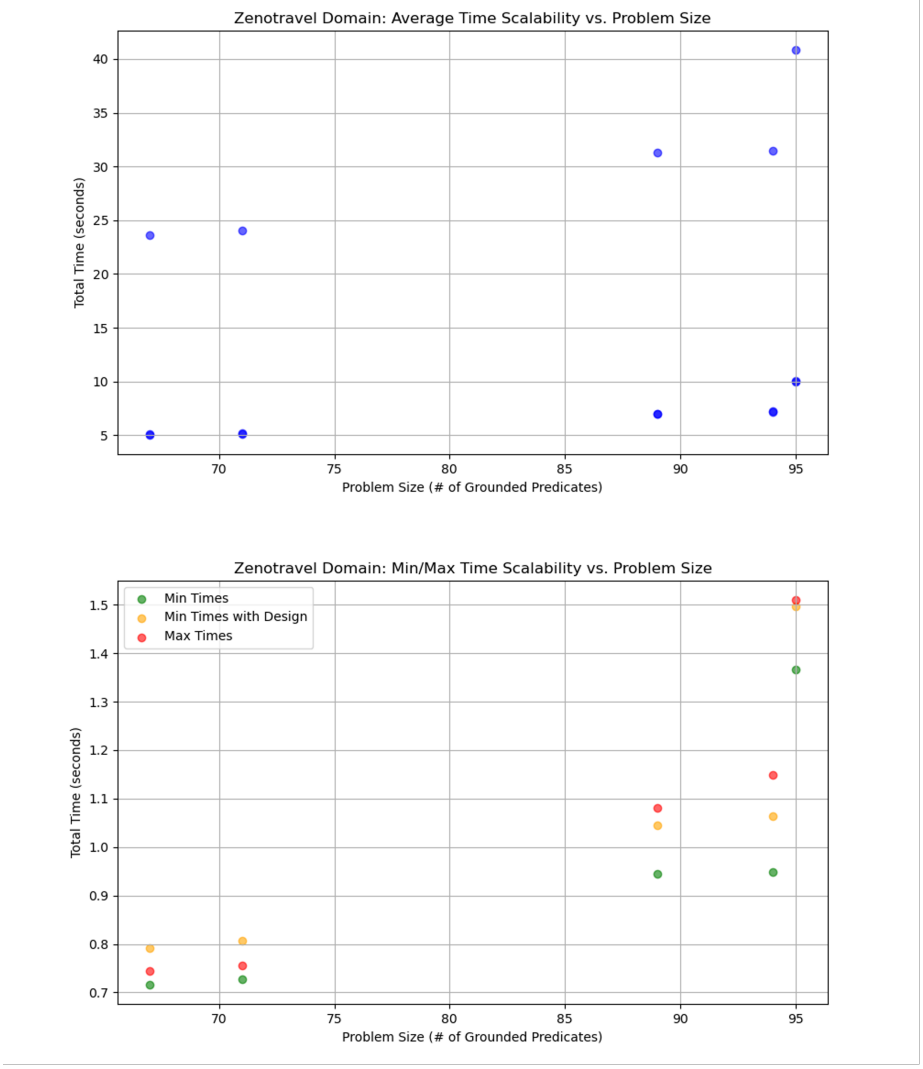
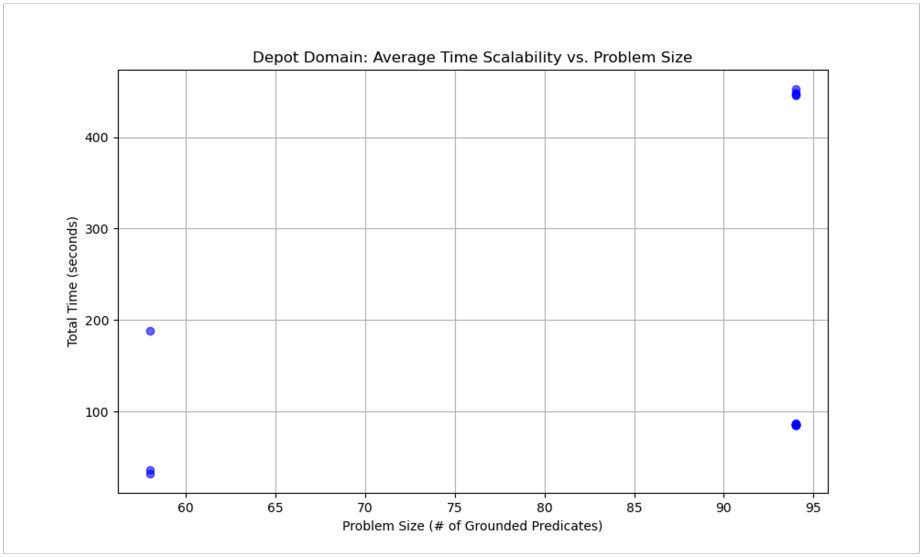


Figure 2: A plot of the time taken for different problem sizes in Zenotravel.



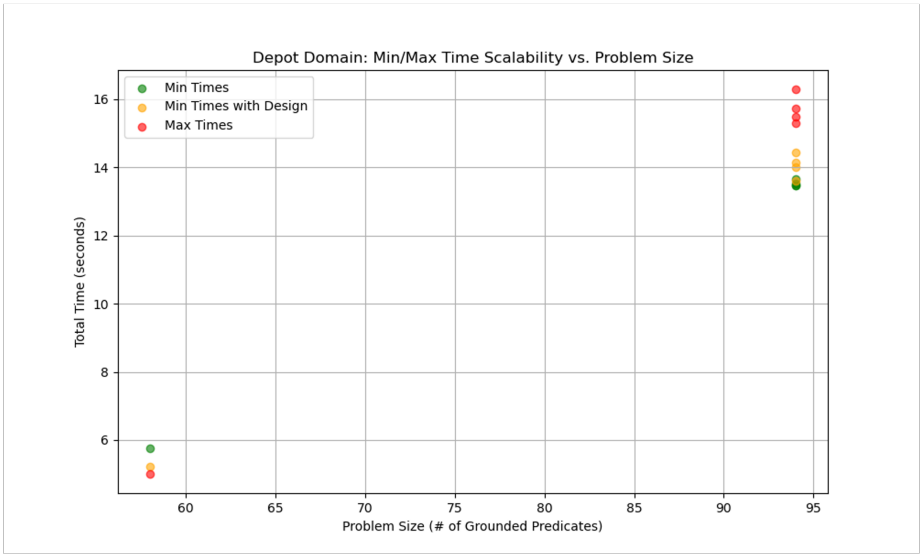


Figure 3: A plot of the time taken for different problem sizes in Depot.

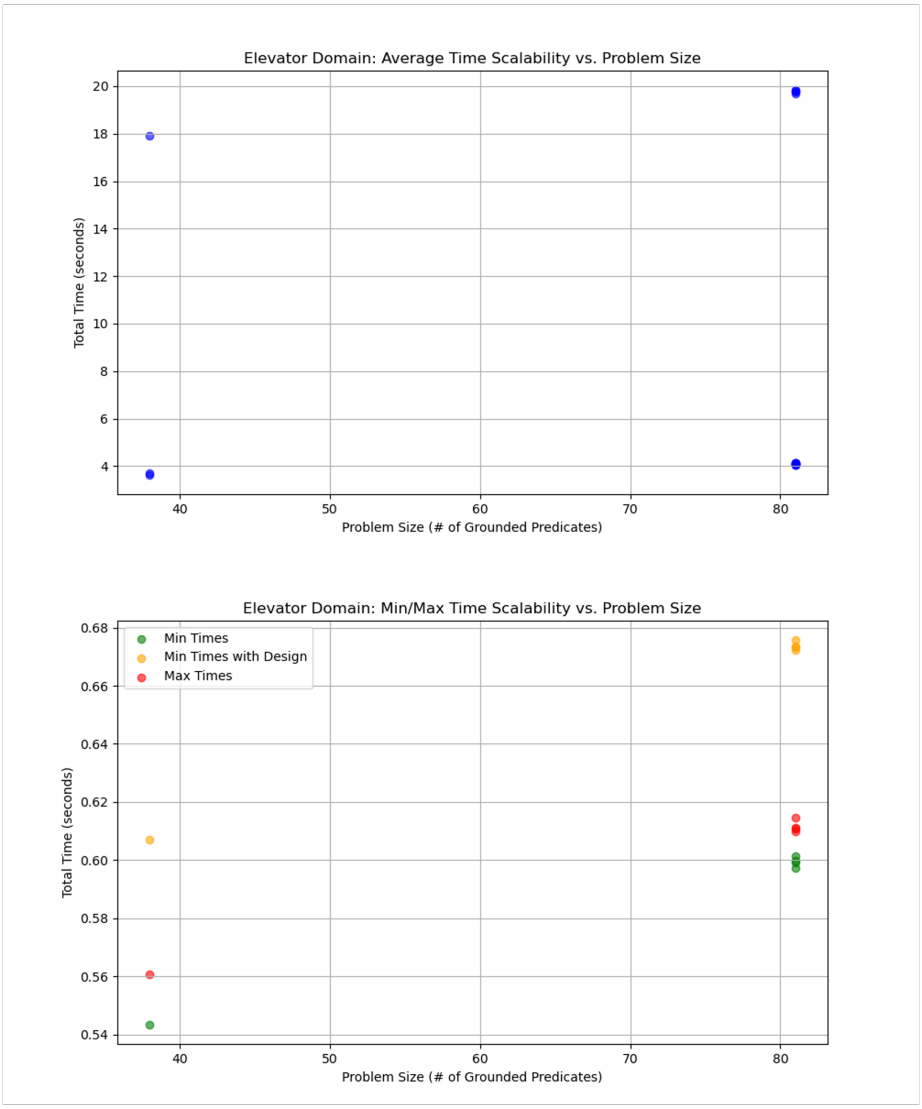


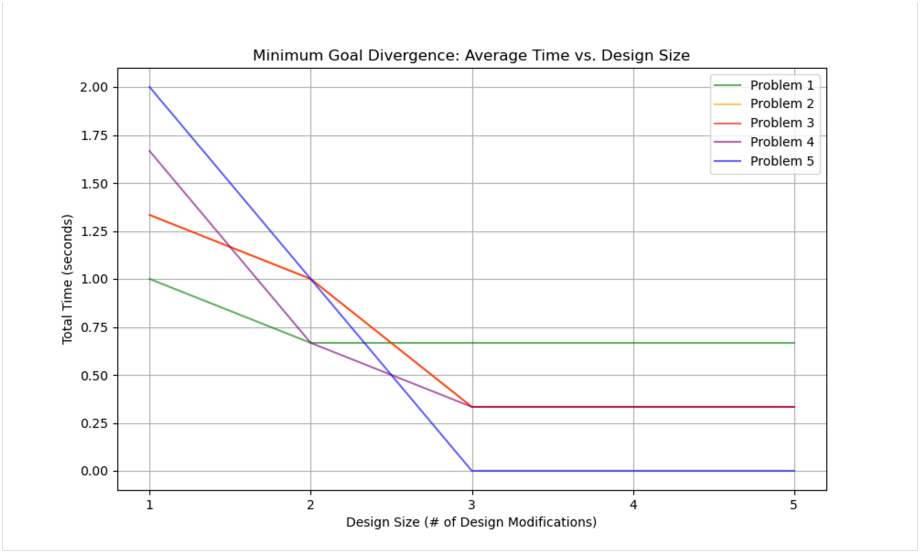
Figure 4: A plot of the time taken for different problem sizes in Elevator.

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

Table 2: The total grounded predicate count per problem instance

4.2 Min and Max GSD Found for Different Design Budget

Next, we were interested in noting how the best case $\mathcal{GD}^{\downarrow}$ and $\mathcal{GD}^{\uparrow}$ evolved as we allowed for larger budgets. In this case, rather than creating problems where we were guaranteed designs that resulted in no goal-state divergence, we wanted to create problems with more flexibility. Here, we still created a human and robot problem copy, by random modifications, but one where we allowed both addition and removal of initial state fluents. We additionally performed a check to see if the problem resulting from the modifications was, in fact, solvable. We focused on a single domain, namely, the Elevator, and created three copies per instance. We then slowly increased the allowed domain modifications from one to five. Then, we noted the best average minimum and maximum divergence we can accomplish within that budget (Figure 5). As expected, we see increasing the budget does result in lower $\mathcal{GD}^{\downarrow}$ and $\mathcal{GD}^{\uparrow}$. We even see that for one of the problems, we were able to achieve zero $\mathcal{GD}^{\downarrow}$ and $\mathcal{GD}^{\uparrow}$.



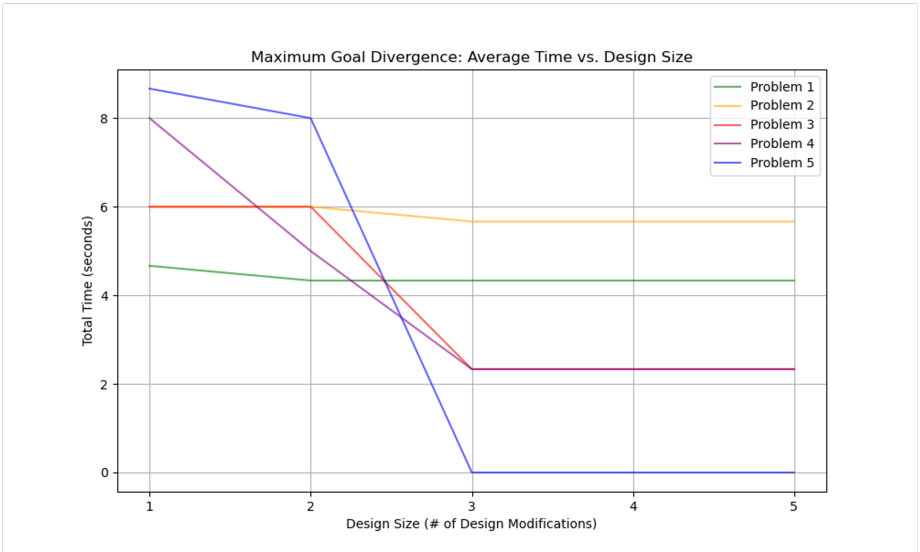


Figure 5: A plot visualizing how the best $\mathcal{GSD}$ approximations change with design budget.

References

Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. In *Proceedings of the international conference on automated planning and scheduling*, volume 20, pages 81–88, 2010.