Information Shaping for Enhanced Goal Recognition of Partially-Informed Agents
Appendix

Introduction

This appendix completes and complements the theoretical discussion presented in our AAAI'20 submission. We start by providing the formal definitions offered by Bonet and Geffner (2011) for the K-planner, which is the solver used by the actor in the goal recognition setting we support, and for the K(P) translation, which is used to transform the actor's planning with partial observability problem into a classical planning problem. We then formally define our $K_{prudent}(P)$ translation and show how it can be used to find an optimistic plan that minimizes the cost as well as the number of assumptions made. Finally, we provide the formal definitions for our pruning technique, CG-Pruning, and specify the conditions under which it can be used to find an optimal design sequence.

The K-planner and the K(P) translation

From a computational perspective, a PPO problem (as Defined in Definition 1) can be formulated as a path finding problem over a graph (Bonet & Geffner, 2000). This approach is common in fully-observable classical planning, only here the nodes of the search graph are belief states, rather then single world states. In this context, solutions approaches to a PPO problem can be characterized as offline or online. In the offline approach, the whole problem is solved and all possible sensing outcomes are considered beforehand. The limitation of the offline approach is that the size of the policies required for solving the problem may have exponential size. A practical alternative is to adopt an online approach that avoids computing a whole policy before acting. Instead, such approaches iteratively select the action to do next, perform the selected action, and observe the result. This happens until either the goal is reached or there are no more actions to perform.

Clearly, online methods are more practical and scale up better than offline methods but do not have the same guarantees (Brafman & Shani, 2012). One common technique for online planning is *replanning* (Zelinsky, 1992), in which the agent finds a plan from its current state based on some simplification of its planning problem. The agent executes some prefix of the plan until new information arrives that requires replanning. In this work we use the K-planner as suggested by Bonet and Geffner (2011), which follows the replanning approach described above. The K-planner constructs a policy that solves a PPO problem P in an on-line fashion, setting the response $\pi(b)$ for the current belief state as the prefix of a plan obtained with an off-the-shelf classical planner. Replanning occurs when new information collected during execution refutes some assumption on which the validity of the executed plan relied.

Algorithm 1 K-planner(PPO problem P)

1: $b_{cur} = b_0$
2: **while** G doesn't hold in b_{cur} (while goal not achieved) **do**
3: $\{\pi_{cur}\} \leftarrow ComputePlan(P, b_{cur})$
4: **if** P has no solution for b_{cur} **then**
5: return { FAIL}
6: **end if**
7: **for** action a in π_{cur} **do**
8: **if** a is not applicable in b_{cur} **then**
9: break (return to Line 2)
10: **end if**
11: $b' \leftarrow APPLY(a, P, b_{cur})$ (apply a to b_{cur}, fire the applicable sensors and update the belief state)
12: **end for**
13: **end while**
14: return { SUCCESS}

The pseudo code for the K-planner is given in Algorithm 1. After initialization of the actor's belief state (Line 1), the iterative planning process, that continues as long as the goal belief has not been reached, starts at Line 2. At each stage, a plan π_{cur} is computed for the current belief state (Line 3). If the current problem has no solution, the solver fails (Line 4). Otherwise, actions in the generated plan are applied in a loop (Line 7). If an action is not applicable (Line 8), re-planing is performed by returning to Line 2. Otherwise, the action is applied and the belief state is updated after

firing applicable sensors and performing inference via the invariant clauses. Line 14 is reached if a goal belief is achieved, in which case 'Success' is returned.

A remaining question deals with the way a plan is generated for the current belief state in Line 3. A practical approach is to generate a plan based on a partial, inaccurate or simplified version of the actual planning problem. In particular, simpler problems can be found by compiling the partially observable planning problem into a classical planning problem obtained from a simplification of the problem (Bonet & Geffner, 2011; ?). Specifically, Bonet and Geffner (2011) use the $K(P)$ translation, defined below.

Definition 1 *For a (PPO) problem* $P = \langle \mathcal{F}, \mathcal{A}, I, G, \mathcal{O} \rangle$, $K(P) = \langle \mathcal{F}', I', G', \mathcal{A}' \rangle$, *is the fully observable problem where*

- $\mathcal{F}' = \{KL, K\neg L : L \in \mathcal{F}\}$
- $I' = \{KL : L \in I\}$
- $G' = \{KL : L \in G\}$
- $\mathcal{A}' = \{\mathcal{A}^{'}_{exe} \cup \mathcal{A}^{'}_{sen} \cup \mathcal{A}^{'}_{ram}\}$ *s.t.*
  - $\mathcal{A}^{'}_{exe} = \mathcal{A}$ *but with each precondition* L *for* $a \in \mathcal{A}$ *replaced by* KL, *and each conditional effect* $C \rightarrow L$ *replaced by* $KC \rightarrow KL$ *and* $\neg K\neg C \rightarrow \neg K\neg L$.
  - $\mathcal{A}^{'}_{sen} = \{a_{(C,L)}, a_{(C,\neg L)} | o = (C, L) \in \mathcal{O}\}$ *where*
    - $prec(a_{(C,L)}) = \{KC, \neg KL, \neg K\neg L\}$ *and* $\mathit{eff}(a_{(C,L)}) = \{KL\}$)
    - $prec(a_{(C,\neg L)}) = \{KC, \neg KL, \neg K\neg L\}$ *and* $\mathit{eff}(a_{(C,\neg L)}) = \{K\neg L\}$
  - $\mathcal{A}^{'}_{ram} = \{a_{ram} | \neg C \vee L \in I\}$ *s.t.*
    - $prec(a_{ram}) = \{KC\}$ *and*
    - $\mathit{eff}(a_{ram}) = \{KL\}$

At the core of the $K(P)$ translation, is the substitution of each fluent L in the original problem with a pair of fluents KL and $K\neg L$, representing whether L is known to be true or false, respectively (Albore, Palacios, & Geffner, 2009). The action set $\mathcal{A}' = \mathcal{A}^{'}_{exe} \cup \mathcal{A}^{'}_{sen} \cup \mathcal{A}^{'}_{ram}$ in the transformed problem contains three types of actions. The set $\mathcal{A}^{'}_{exe}$ represents the original set of actions such that each original action $a \in \mathcal{A}$ is transformed into an equivalent action that replaces the use of every literal L ($\neg L$), with its corresponding fluent KL ($K\neg L$). The set $\mathcal{A}^{'}_{sen}$ represents the observations that may be collected during execution. Each observation (C, L) is translated into two deterministic sensing actions, one for each possible value of L. These sensing actions allow the solver to compute a plan while choosing preferred values of (making assumptions about) unknown variables. Finally, the set $\mathcal{A}^{'}_{ram}$ are what we refer to as *ramification actions*, which represent the inference process performed by the agent. Each invariant clause defined in I is translated into a set of ramification actions that can be used to set the truth value of some variable, as new sensed information is collected from the environment. This representation captures the underlying planning problem at the knowledge level (Petrick & Bacchus, 2002), accounting for the exploratory behavior of a partially informed agent.

By using a transformation to classical planning, it is possible to use any off-the-shelf classical planner to compute a plan to the transformed problem. Bonet and Geffner (2011) show that the linear K(P) translation is sound and complete for simple problems.

The K-planner together with the K(P) transformation allow the actor's planner to follow a *planning under optimism* approach; the actor plans while making the most convenient assumptions about the values of hidden variables, executes the plan that is obtained from the resulting classical planning problem, and revises the assumptions and re-plans, if during the execution, an observation refutes the assumptions made.

The $K_{prudent}(P)$ Translation

A key issue to note about the $K(P)$ translation by Bonet and Geffner (2011) (Definition 1) is that all the actions in the compiled planning problem, including sensing and ramification actions, have equal cost. This means that solutions to the resulting classical planning problem do not distinguish between the different types of actions in $\mathcal{A}'$. This means that an optimal plan to $K(P)$ may be one that favors, for example, increasing the number of execution actions over the use of multiple ramification actions.

We want a solver that makes optimistic (i.e. convenient) assumptions, but chooses a cost-minimal plan that requires making as few assumptions as possible. In addition, ramifications, which represent reasoning about available information, should not to be considered when calculating the cost to goal. We therefore suggest the $K_{prudent}(P)$ translation, formally defined in Definition 2, which extends the uniform cost $K(P)$ translation by associating a cost function to each action in $\mathcal{A}'$.

Definition 2 *Given a PPO problem* $P = \langle \mathcal{F}, \mathcal{A}, I, G, \mathcal{O} \rangle$, *the* $K_{prudent}(P)$ *is defined from* $K(P) = \langle \mathcal{F}', I', G', \mathcal{A}' \rangle$ *as the fully observable problem* $P'' = \langle \mathcal{F}', I', G', \mathcal{A}', \mathcal{C}'' \rangle$ *where*

$$\mathcal{C}''(a) = \begin{cases} 1 & \text{if } a \in \mathcal{A}^{'}_{exe} \\ \epsilon & \text{if } a \in \mathcal{A}^{'}_{sen} \\ 0 & \text{if } a \in \mathcal{A}^{'}_{ram} \end{cases}$$

According to our $K_{prudent}(P)$ translation, every transformed action $a \in \mathcal{A}^{'}_{exe}$ is assigned a cost of 1, every sensing action (assumption) $a \in \mathcal{A}^{'}_{sen}$ is assigned a cost of ϵ, and every ramification action $a \in \mathcal{A}^{'}_{ram}$ has 0 cost.

Our objective is to specify conditions under which an optimal plan to $K_{prudent}(P)$ is one that minimizes

execution cost, while making the minimal number of assumptions.

First, we define a plan as the *Optimistic − Optimal* plan of a PPO problem $\mathcal{P}$ if minimizes the number of execution actions that are performed to reach a goal belief from the initial belief.

Definition 3 (Optimistic-Optimal cost and plan) *Given a PPO problem $\mathcal{P}$, the* Optimistic-Optimal *cost of $\mathcal{P}$, denoted by $C^{OpOp}(\mathcal{P})$, is the minimum among the costs of plans that reach a goal belief from a state in the initial belief b_0. A plan π is an* Optimistic-Optimal *plan of $\mathcal{P}$ if it has the minimum cost among the plans that reach a goal belief from a state in the initial belief b_0.*

Our objective is to specify conditions under which an optimal plan to $K_{prudent}(P)$ is an Optimistic-Optimal plan which requires making the minimal number of assumptions. To formulate this requirement, we let $\pi_{P''}$ represent a plan found by a planner for the compiled problem P'', and let $\pi^*_{P''}$ represent a solution found by an optimal planner. In addition, we let $C''_{exe}(\pi_{P''})$ represent the plan's *execution cost*, i.e., the accumulated cost of execution actions ($\mathcal{A}'_{exe}$) in π, which are the only actions in A' that correspond to actions in the original problem P. We let $\Pi''_{exe}(\pi)$ represent the **execution projection** of π, the sequence of execution actions of π (excluding ramification and sensing actions). Equivalently, $C''_{sen}(\pi)$ is the accumulated cost of sensing actions $\mathcal{A}'_{sen}$, corresponding to the assumptions made as part of the plan (we do not consider the cost of ramification actions since they all have 0 cost).

We specify our requirement that $\pi^*_{P''}$ is an Optimistic-Optimal plan of P which includes the minimal number of assumptions, represented by actions in $\mathcal{A}'_{sen}$. Specifically, we show that when ϵ is small enough such that the accumulated cost of assumptions of an optimal solution to P'' is guaranteed to be smaller than minimal diversion from an optimal plan, $\pi^*_{P''}$ is a plan that satisfies our requirement.

Theorem 1 *Given a PPO problem $P = \langle \mathcal{F}, \mathcal{A}, I, G, \mathcal{O} \rangle$, and its $K_{prudent}(P)$ translation P'', if $\epsilon < \frac{2}{|\mathcal{A}'_{sen}|}$ then*
(1) $C''_{exe}(\pi^*_{P''}) = C^{OpOp}(P)$ and
(2) for any other plan π of P'' s.t. $C''_{exe}(\pi) = C^{OpOp}(P)$ then $C''_{sen}(\pi^*_{P''}) \leq C''_{sen}(\pi)$.

Proof:

We start by proving that $C''_{exe}(\pi^*_{P''}) = C^{OpOp}(P)$. Since by definition $C^{OpOp}(P)$ is the minimal cost of a path from any state in the initial belief to a goal belief, $C''_{exe}(\pi^*_{P''})$ cannot be smaller then $C^{OpOp}(P)$. We therefore assume to the contrary that $C''_{exe}(\pi^*_{P''}) > C^{OpOp}(P)$.

Since the minimal difference between execution costs of two plans is 1, we know that

$$C''_{exe}(\pi^*_{P''}) - C^{OpOp}(P) > 1$$

W.l.o.g, we let plan π represent a plan in P'' s.t. $C''_{exe}(\pi) = C^{OpOp}(P)$. According to our assumption, since $\pi^*_{P''}$ is optimal for P''

$$C''_{exe}(\pi^*_{P''}) + C''_{sen}(\pi^*_{P''}) \leq C^{OpOp}(P) + C''_{sen}(\pi)$$

Since $C''_{exe}(\pi^*_{P''}) - C^{OpOp}(P) > 1$ this means that $C''_{sen}(\pi) \geq 1$. However, the maximal number of assumptions that can be made is $\frac{|\mathcal{A}'_{sen}|}{2}$, and the maximal cost that can be incurred for making assumptions is $\frac{\epsilon \cdot |\mathcal{A}'_{sen}|}{2}$, therefore

$$\frac{\epsilon \cdot |\mathcal{A}'_{sen}|}{2} \geq 1$$

requiring that

$$\epsilon \geq \frac{2}{|\mathcal{A}'_{sen}|}$$

which contradicts our assumption, guaranteeing that $\pi^*_{P''}$ has minimal execution cost.

Moreover, the only cost difference between the plans with minimal execution cost is due to the cost of $C''_{sen}(\pi)$ of assumption, which guarantees that among the plan with minimal execution cost, an optimal planner will select a plan that minimizes the number of assumption (actions in $\mathcal{A}'_{sen}$), thus concluding our proof. ∎

Methods for Design

The challenge in information shaping comes from two sources. First, the number of possible information shaping options may be large, and evaluating the effect of each change may be costly, making it important to develop efficient search techniques. Second, in many cases the problem is non-monotonic, in that sensor extensions are not always useful, and providing more information to the actor may actually decrease the system's utility. This is true specifically to the two special cases we are considering here, ER-UMD-APK and GRD-APK, both presented in the previous section.

Design as State Space Search

To address these challenges, we follow Keren, Gal, and Karpas (2018) and formulate the design process as a search in the space of modification sets $\Delta \subseteq \mathbf{\Delta}$. With a slight abuse of notation, we let R^{Δ} denote the model that results from applying the set Δ of sensor extensions to the actor's sensor model. The root node is the initial goal recognition model R_0 (and empty modification set), and the operators (edges) are the sensor extensions $\delta \in \mathbf{\Delta}$ that transition between models. Each node (modifications set Δ) is evaluated by $WCD(R_0^{\Delta})$, the *WCD* value of its corresponding model. Figure 1

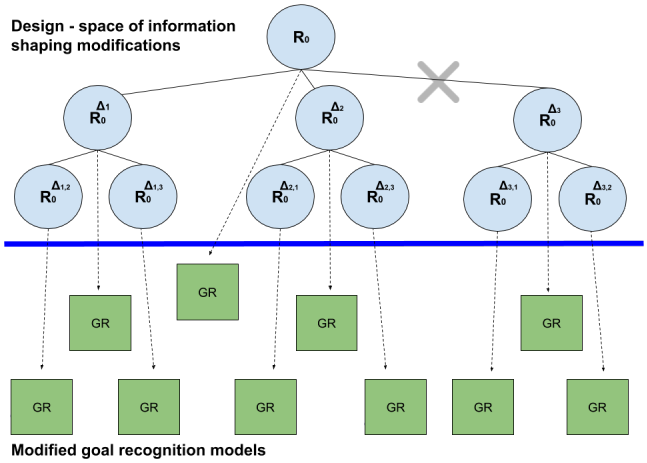

Figure 1: Information Shaping

depicts the search in the design space, where nodes on the top are the design nodes, corresponding to sets of sensor extensions. Each design node corresponds to a modified goal recognition setting at the bottom of the figure.

The baseline approach for searching in the modification space is *breadth first search* (BFS), using the targeted utility measure to evaluate each node. Under the budget constraints, BFS explores modification sets of increasing size, using a closed-list to avoid the computation of pre-computed sets. The search halts when there are no more nodes to explore. When a bound on the utility value is known, the search halts when if a model with the maximal (or minimal) utility value is found. For example, for GRD-APK the search halts when a node with $WCD = 0$ is found. The returned value is the shortest path (smallest modification set) to a node that achieves the maximal value. This iterative approach is guaranteed to find a strongly optimal solution, i.e., a minimal set of modifications that maximize the utility. However, it does not scale to larger problems. [1]

Design with CG-Pruning

To support efficiency, pruning can be applied to reduce the size of the search space. Specifically, pruning is *safe* if at least one optimal solution remains unpruned (Wehrle & Helmert, 2014). Keren et al. (2018) offer a pruning technique for goal recognition design (GRD) settings where the actor is fully informed and guarantee it is safe if modifications cannot increase *WCD*. As we demonstrate in Example **??**, this condition does not hold in our setting, where sensor extensions can both increase and decrease utility. We therefore suggest a new approach that eliminates useless modifications, and specify conditions under which it is safe.

The high level idea of our pruning technique is to transform the partially observable planning problem for each goal into its corresponding fully observable planning problem, and use off-the-shelf tools developed for fully observable planning in order to automatically detect information shaping modifications that are guaranteed not to have an effect on the actor's behavior.

Specifically, given a model R, for every goal in $\mathcal{G}$, we use the $K(P)$ transformation (or its variant $K_{prudent}(P)$ introduced above) to transform the partially observable planning problem into a fully observable problem. We then construct the *causal graph* (Williams & Nayak, 1997; Helmert, 2006) of each transformed problem. According to Helmert (2006), the causal graph of a planning problem is a directed graph (V, E) where the nodes V represent the state variables and the edges E represent dependencies between variables, such that the graph contains a directed edge (v, v') for $v, v' \in V$ if changes in the value of v' can depend on the value of v. Specifically, to capture only the variables that are relevant to achieving the goal, the causal graph only contains ancestors of all variables that appear in the goal description.

Causal Graphs Below we provide the formal definition of the causal graph and the domain transition graph on which it is based. Note that the original formulation of causal graphs by Helmert (2006) accounted for multi-valued planning tasks that included axioms (invariants) over the set of variables, which are formulations for extracting derived variable. As we have shown above, in our contingent planning models axioms are captured in the initial state of the original problem and are transformed into ramification actions ($\mathcal{A}'_{ram}$) of the compiled problem. We therefore exclude them from the definitions given below.

A *domain transition graph* is a representation of the ways variable values can change, and of the conditions that must be satisfied for such value changes to be possible.

Definition 4 (Domain Transition Graph) *Let $\Pi = \langle \mathcal{V}, s_0, s_g, O \rangle$ be a multi-valued planning task, and let $v \in \mathcal{V}$ be a state variable of π.*

The domain transition graph *of v, in symbols $DTG(v)$, is a labeled directed graph with vertex set $\mathcal{D}_v$. If v is a fluent, $DTG(v)$ contains the following arcs:*

- *For each effect $cond \rightarrow v := d'$ of an operator o with precondition $prec$ such that $prec \cup cond$ contains some condition $v = d$, an arc from d to d' labeled with $prec \cup cond \setminus \{v = d\}$.*
- *For each effect $cond \rightarrow v := d'$ of an operator o with precondition $prec$ such that $prec \cup cond$ does not con-*

[1] We note that in the case of non-uniform modification costs, we are interested in a modification sequence with minimal cost. In this case, we can replace the BFS with a Dijkstra-based exploration. Instead of the queue described above, the sequences are maintained in a priority queue that returns at each stage the minimal-cost sequence.

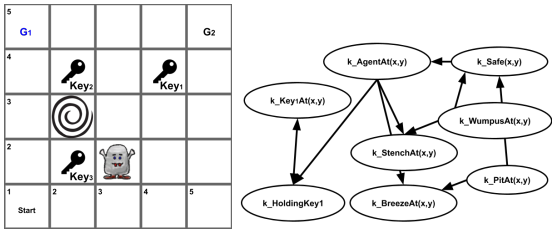

Figure 2: The Wumpus domain with keys

tain the condition $v = d$ for any $d \in \mathcal{D}_v$, an arc from each $d \in D_v \setminus d'$ to d' labelled with pre $\cup$ cond.

For a variable v, the domain transition graph contains a transition from d to d' if there exists some operator that changes the value of v from d to d'. The labels capture the connections between a variable and other state variables that must be true for some value change to be happen. The causal graph relies on the domain transitions graph to capture these connections.

Definition 5 (Causal Graph) *Let Π be a multi-valued planning task with variable set $\mathcal{V}$. The causal graph of Π, in symbols $CG(\Pi) = (V, E)$, is the directed graph with vertex set $\mathcal{V}$ containing an arc (v, v') iff $v \neq v'$ and one of the following conditions is true:*

- *The domain transition graph of v' has a transition with some condition on v.*
- *The set of affected variables in the effect list of some operator includes both v and v'.*

In the first case, we say that an arc is induced by a transition condition. *In the second case we say that it is induced by* co-occurring effects.

To capture only the variables that are relevant to achieving the goal, the causal graph only contains ancestors of all variables that appear in the goal description.

Our causal graph analysis considers the classical planning problem that results from transforming a PPO problem P using either $K(P)$ or $K_{prudent}(P)$ transformation to classical planning. Since the causal graph analysis ignores costs, our analysis here applies to both approaches, so we will hereon use the former to refer to the transformation used.

A goal recognition problem R induces a separate planning problem for each goal. In turn, the causal graph of the $K(P)$ transformation for each goal $G \in \mathcal{G}$, captures the relationships between variables in the transformed problem. Note that for each goal $G \in \mathcal{G}$ and transformed problem $P' = \langle \mathcal{F}', I', G', \mathcal{A}' \rangle$, the variable set of the causal graph can either be the set of fluents $\mathcal{F}'$ of the transformed PPO problem, or a set of multi-valued variables extracted using *invariant synthesis*, which automatically finds sets of fluents among which exactly one is true at each state. We let $\mathcal{V}'$ denote the set of mult-valued for P'.

Since $K(P)$ includes three types of actions, an edge connecting two variables in $\mathcal{V}'$ can have one of three meanings. If the connection (v_i, v_j) is labeled by an action in A_{exe}, the knowledge about the value of v_i affects the applicability of an action that changes the value of v_j. If the connection is labeled by an action in A_{sen}, knowledge about the value of v_j can be observed when the value of v_i is known. Finally, it A_{ram} appears in the label, then the value of the v_j can be deduced when the value of v_i is known. In this way, the causal graph captures the relationships between variables at the knowledge level.

Consider Figure 2(left), depicting a modified version of Example 1, where the actor needs to collect a key to be able to access its goal (e.g., $PickedKey_1$ is needed to reach G_1). There are multiple keys distributed in the grid (e.g., $Key_1At(4,4)$), each needed for accessing a particular location. The actor initially knows a set of possible key locations for each key. When in a cell with a key, it senses it and can pick it up and use it to achieve its goal. In this scenario, the recognizer, with perfect information, can notify the actor about safe locations, as before, but also about the absence or presence of a particular key in some location. Applying the $K(P)$ transformation here creates fluents $KKey_iAt(x, y)$ for each key and location, representing whether the actor knows key i is at location (x, y), which is a precondition to picking up the key. Figure 2(right), shows a part of the causal graph for G_1 that only includes variables concerning the location of its relevant key. By generating the causal graph to all goals, we automatically detect and prune sensor extensions regarding variables that do not appear in any of the causal graphs (e.g., the sensor extension that reveals the location of Key_3).

Safe Pruning with CG-Pruning We are exploring the relationship between sensor extensions and the actor's behavior, and are seeking an automated way to identify sensor extensions that will not affect the *WCD* value of a model. To implement this, we are using the causal graphs of the actor's planning problem, which can be automatically extracted from the actor's planning problem.

Under the assumption that our actor uses the *k-planner* to decide how to act, it iteratively computes a plan at the initial state and every time an assumption made at a previous iteration is refuted. At each iteration, the current partially observable problem is trans-

formed into its corresponding fully observable problem, and a new plan is computed and executed. This continues until the actor reaches a goal belief or a belief state with no applicable actions.

For each goal recognition model R, goal $G \in \mathcal{G}$ and execution iteration i, we let $CG_i^R(G)$ represent the causal graph that corresponds to the actor's planning problem at iteration i. Recall that a sensor extension δ adds some observation $o = (C, L)$ to the actor's sensor model. In the following, we say that a sensor extension is *relevant* to causal graph $CG_i^R(G)$, marked $L \in CG_i^R(G)$, if L or $\neg L$ appears in the variables of $CG_i^R(G)$. Note that since L is a fluent and the causal graph may contain multi-valued variables extracted from the fluents of the problem, L is relevant to $CG_i^R(G)$ if either L or $\neg L$ appear as a node in $CG_i^R(G)$ or as a value in the domain of one its multi-valued variables. CG-Pruning prunes sensor extensions not related to $CG_0^R(G)$ for any goal $G \in \mathcal{G}$.

In order to specify conditions under which CG-Pruning is safe, we assume our actor uses the k-planner with an optimal planner used to compute a plan for the $K(P)$ (or $K_{prudent}(P)$) transformation of its current planning problem. As detailed below, our proof relies on the fact that a sensor extension that is not relevant to the causal graph of an actor's planning problem will not change the actor's behavior. In addition, variables that do not appear in the initial causal graph will not appear in causal graphs of future iterations. We can therefore safely prune modifications that do not appear in any of the causal graphs since they are guaranteed not to be useful.

We start our proof by showing that the causal graph at each iteration subsumes causal graphs of subsequent iterations.

Lemma 1 *For any model R and goal $G \in \mathcal{G}$, $CG_j^R(G)$ is a subgraph of $CG_i^R(G)$ for any i, j s.t. $0 \leq i < j$.*

Proof:

Assume to the contrary that exists $G \in \mathcal{G}$ and i, j $0 \leq i < j$ s.t. $CG_j^R(G) = (V_j, E_j)$ is not a subgraph of $CG_i^R(G) = (V_i, E_i)$. This means that either $\exists v \in V_j$ s.t. $v \notin V_i$. or $\exists e \in E_j$ s.t. $e \notin E_i$.

According to Definition 1, for every goal G, the transformed problem at iteration i is $K(P)_i = \langle \mathcal{F}'_i, I'_i, G'_i, \mathcal{A}'_i \rangle$ while the problem at iteration j is $K(P)_j = \langle \mathcal{F}'_j, I'_j, G'_j, \mathcal{A}'_j \rangle$. According to Definition 5, the variables of the causal graph correspond to the variables of the planning problem. Since $\mathcal{F}'_i = \mathcal{F}'_j$ then $V_i = V_j$, so the first condition cannot occur, which leaves us with the need to refute the assumption that $\exists e \in E_j$ s.t. $e \notin E_i$.

Since the problem's fluents, actions and goals don't change as a result of the actor's progress ($\mathcal{F}'_i = \mathcal{F}'_j$, $\mathcal{A}'_i = \mathcal{A}'_j$ and $G'_i = G_j$), the only potential source of difference between $CG_j^R(G)$ and $CG_i^R(G)$, is the initial belief state at each iteration. However, according to Definition 5, the initial state does not affect the generated causal graph, and therefore the graphs are the same and any edge in E_j appears in E_i, thus contradicting our assumption and concluding our proof. ∎

We now show that when an optimal solver is used, a sensor extension that is not relevant to the actor's causal graph of any goal is not useful.

Lemma 2 *For any model R and sensor extension δ, if δ is not relevant to $CG_0^R(G)$ for any goal $G \in \mathcal{G}$, then δ is not useful w.r.t R.*

Proof: Bonet and Geffner (2011) show that the $K(P)$ transformation is sound and complete for simple problems with a connected space, which are the only problems we consider here. Helmert (2006) shows that any optimal plan can be acquired by ignoring variables that are not in the causal graph. Therefore, by pruning sensor extensions that are not relevant to a causal graph, we are removing from the actor's planning graph sensing actions that would anyway not appear in any optimal plan (i.e., assumptions the actor would not make). Therefore, the behavior of an actor to any goal is not affected by such sensor extensions. Moreover, Lemma 1 guarantees that if a sensor extension is not related to $CG_0^R(G)$, it will not be related to any $CG_i^R(G)$, for $0 < i$. Consequently, if δ is not related to $CG_0^R(G)$ will not change the behavior of the actor w.r.t to any goal, *WCD* will not change, and therefore δ is not useful w.r.t R. ∎

Finally, we are ready to show that CG-Pruning is safe.

Theorem 2 *For any GRD-APK model $T = \langle R_0, \mathbf{\Delta}, \beta \rangle$, CG-Pruning is safe for an actor that uses the* k-planner *with an optimal planner.*

Proof: Lemma 2 shows that under the assumptions we make about the actor, any sensor extension δ not related to $CG_0^R(G)$ for any goal $G \in \mathcal{G}$ is not useful w.r.t R. This means that such extensions are guaranteed not change the *WCD* and are not a part of an optimal solution and can therefore be pruned. ∎

References

Albore, A., Palacios, H., & Geffner, H. (2009). A translation-based approach to contingent planning.. In *IJCAI*.

Bonet, B., & Geffner, H. (2000). Planning with incomplete information as heuristic search in belief space. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS)*, pp. 52–61. AAAI Press.

Bonet, B., & Geffner, H. (2011). Planning under partial observability by classical replanning: Theory and experiments. In *IJCAI*.

Brafman, R. I., & Shani, G. (2012). Replanning in domains with partial information and sensing actions. *Journal of Artificial Intelligence Research*, *45*, 565–600.

Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research(JAIR)*, *26*.

Keren, S., Gal, A., & Karpas, E. (2018). Strong stubborn sets for efficient goal recognition design. In *ICAPS*.

Petrick, R. P., & Bacchus, F. (2002). A knowledge-based approach to planning with incomplete information and sensing.. In *ICAPS*.

Wehrle, M., & Helmert, M. (2014). Efficient stubborn sets: Generalized algorithms and selection strategies. In *ICAPS*.

Williams, B. C., & Nayak, P. P. (1997). A reactive planner for a model-based executive. In *IJCAI*, Vol. 97.

Zelinsky, A. (1992). A mobile robot exploration algorithm. *IEEE transactions on Robotics and Automation*, *8*(6), 707–717.