Information Shaping for Enhanced Goal Recognition of Partially-Informed Agents

Abstract

We extend goal recognition design to account for partially informed agents. In particular, we consider a two-agent setting in which one agent, the *actor*, seeks to achieve a goal but has only incomplete information about the environment. The second agent, the *recognizer*, has perfect information and aims to recognize the actor's goal from its behavior as quickly as possible. As a one-time offline intervention and with the objective of facilitating the recognition task, the recognizer can selectively reveal information to the actor. The problem of selecting which information to reveal, which we call *information shaping*, is challenging not only because the space of information shaping options may be large, but also because more information revelation need not make an agent's goal easier to recognize. We formally define this problem, and suggest a pruning approach for efficiently searching the search space. We demonstrate the effectiveness and efficiency of the suggested method on standard benchmarks.

Introduction

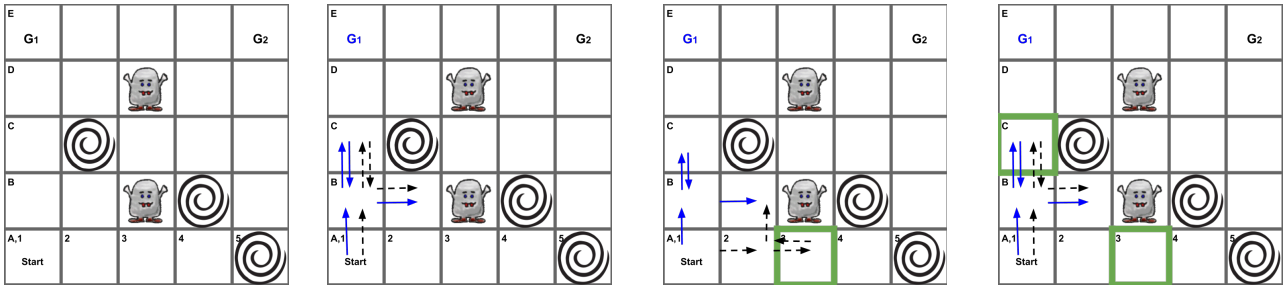
Goal recognition is the task of detecting the goal of agents by observing their behavior (Cohen, Perrault, and Allen 1981; Kautz and Allen 1986; Ramirez and Geffner 2010; Carberry 2001; Sukthankar et al. 2014). We consider a two-agent goal recognition setting, where the first agent, the *actor*, has partial information about a deterministic environment and seeks to achieve a goal. The second agent, the *recognizer*, has perfect information, and tries to deduce the actor's goal as early as possible, by analyzing the actor's behavior.

As a one time offline intervention, and with the objective of facilitating the recognition task, the recognizer can apply a limited number of *information shaping* modifications, implemented as changes to the actor's sensor model. Such modifications can potentially help to differentiate the actor's behavior for different goals, making it easier to interpret.

The ability to quickly understand what an agent is trying to achieve, without expecting it to explicitly communicate its objectives, is important in many applications. For example, in an assistive cognition setting (Kautz et al. 2003), it may be critical to know as early as possible when a visually impaired user (i.e., the actor) is approaching a hot oven, giving the system (i.e., recognizer) time to react to the dangerous situation (e.g., by calling for help, reducing the heat, etc.). In security applications it may be important to early detect users aiming at a specific destination (Boddy et al. 2005), giving the system enough time to send human agents to further investigate potential threats. Early detection is also important in human-robot collaborative settings (Levine and Williams 2014), where a robot aims to recognize what component a human user is trying to assemble, so it can gather the tools needed for the task in a timely fashion. Common to all these settings, is that actors have *incomplete information* about their environment. This affects their behavior and is key to the ability to interpret it. In addition, these settings can be controlled and modified in various ways. Specifically, it may be possible to modify an agent's behavior by manipulating its knowledge and its need to act in order to acquire new information. Such manipulations may induce behaviors that can be quickly associated to a specific goal. To demonstrate, in an assisted cognition setting an auditory signal can inform users about a hot oven. Early notification potentially causes users aiming at a different goal (e.g., the cupboard) to move away from the oven, supporting early recognition of dangerous situations.

This work extends the *goal recognition design* (GRD) framework, which deals with redesigning agent settings in order to facilitate early goal detection (Keren, Gal, and Karpas 2014; Wayllace et al. 2016). Until now, GRD work has assumed the actor has perfect knowledge of the environment. In this paper, we extend the framework to support actors with incomplete knowledge. Specifically, we focus on GRD in deterministic environments, and use contingent planning (Bonet and Geffner 2011; Brafman and Shani 2012a; Muise, Belle, and McIlraith 2014; Albore, Palacios, and Geffner 2009) to represent the actor. The design objective is to minimize *worst case distinctiveness* (*WCD*) (Keren, Gal, and Karpas 2014), which represents the maximal progress an actor can make in the system before the recognizer can deduce its goal. Note that in some instances the goal may remain unrecognized, and even go unattained, in which case we consider the number of actions (or accumulated action cost) until the end of execution.

To minimize *WCD* we introduce *information shaping* as a way to induce desired behaviors, but require that the information conveyed to the actor is truthful and cannot mis-



(a) The goal recognition setting: an actor to either goal may move up.

(b) Plans executed by an actor aiming at G_1 (solid arrows) and G_2 (dashed arrows).

(c) Plans for each goal when the recognizer reveals (3,1) is safe

(d) Plans for each goal when the recognizer reveals both (3,1) and (1,3) are safe.

Figure 1: An example of a GRD-APK problem

lead. Specifically, we use *sensor extensions* to selectively facilitate the actor's access to information about the value of some environment variables. This is a challenging problem not only because the number of possible design options may be extremely large, but also, as we demonstrate below, because the problem is *non-monotonic* in that more information need not make an actor's goal easier to recognize.

Example 1 *Consider Figure 1(a), depicting a variation of the Wumpus domain (Russell and Norvig 2016), where a partially informed actor has one of two goals (indicated by G_1 and G_2), and needs to achieve the goal without falling into a pit or encountering a deadly wumpus. The actor knows its current position, but initially does not know the locations of the pits and wumpuses. When in a cell adjacent to a pit or wumpus, it senses a 'breeze' or 'stench', respectively, without knowing which direction the signal originated from. The recognizer has perfect information: it knows the locations of the actor, the pits (e.g., the spiral at cell $(2, C)$) and the wumpuses (e.g., cell $(3, B)$), but not the actor's goal.*

We assume the actor is optimistic when planning but conservative when acting (Bonet and Geffner 2011). Optimistic planning means the planner is willing to make assumptions about unknown variables under which it can follow a cost-minimal plan to the goal. It is conservative in that it only performs actions for which it knows the outcome. The planner we present in this work further tie-breaks among optimal plans in favor of making as few assumptions as possible, and arbitrarily otherwise. During execution, the actor collects observations from the environment and revises its assumptions and re-plans if an assumption made is refuted.

The actor enters the system at 'Start'. With no breeze or stench, it decuces the adjacent cells are safe. *An actor aiming at G_1 will start by moving up. An actor aiming at G_2 is indifferent to going up or right, and may go either way. Because of this, moving up leaves the goal unrecognized. Let us suppose (Figure 1b) the actor initially assumes all cells in the left column are safe, and that plans to both goals start by moving up two steps. After sensing a breeze at cell (1,C), not knowing which adjacent cells have a pit, the actor backtracks and moves right. After sensing a 'breeze' and 'stench', the actor deduces there is a wumpus at cell (3,B), and realizes that it will sense a stench at cell (3,A), without having the option of verifying that cell (4,A) is safe. With no more cells to explore, it halts at (2,B) leaving the goal unrecognized even after it terminates execution (*WCD$= 4$*).*

The recognizer can share information, for example by revealing safe cells. If the recognizer chooses to reveal cell $(3, A)$ is safe (Figure 1(c)), an actor aiming at G_2 now prefers moving right from the initial state while an actor aiming at G_1 still prefers moving up. The goal of the actor becomes clear as soon as the first step is performed and WCD$= 0$. *However, if the recognizer also reveals that cell $(1, C)$ is safe (Figure 1(d)), the initial situation is recovered, since an actor to either goal may now choose to move up given its beliefs. Moreover, note that if communication is unlimited and the recognizer chooses to provide complete information to the actor* WCD $= 4$. *This illustrates non-monotonicity and the need to carefully select the information to reveal in order to facilitate recognition.*

The contributions of this work are four-fold. First, we extend the GRD framework to support agents with partial information. We refer to our extended setting as *GRD for Agents with Partial Knowledge* (GRD-APK), and introduce information shaping that can be applied to support goal recognition. Second, we introduce a new solver for planning under partial observability that supports an actor that can make assumptions about unknown variables, but prefers to make as few assumptions as possible. Third, since our design setting induces a large search space of possible information shaping modifications and since previous approaches do not apply to our non-monotonic setting, we present a novel pruning method that uses off-the-shelf classical planning tools to automatically detect useless information shaping options. We specify the conditions under which our pruning is *safe*, so that at least one optimal solution is not pruned. Finally, to evaluate our approach, we structure information shaping as a planning problem and implement it using STRIPS (Fikes and Nilsson 1972) to represent our general and adaptable redesign process. We demonstrate *WCD* reduction achievable through information shaping and the efficiency of our algorithm on a set of standard benchmarks.

Related Work

Goal Recognition Design (GRD), a special case of *environment design* (Zhang, Chen, and Parkes 2009), was first introduced by Keren et al. (2014) to account for optimal

fully observable agents in deterministic domains. This work was later extended to a variety of GRD settings, including accounts for sub-optimal actors (Keren, Gal, and Karpas 2015), stochastic environments (Wayllace et al. 2016), adversarial actors (Ang et al. 2017), and a partially informed recognizer (Keren, Gal, and Karpas 2016a; 2016b; 2018). In the latter case, sensor refinement is applied to enhance the recognizer's sensor model.

Common to all previous GRD work is the assumption that actors have perfect observability of their environment. Our work is the first to generalize GRD to account for a partially informed actor and to suggest new information shaping modifications, implemented as sensor extensions applied to the actor's sensor model, as a way to reduce *WCD*.

Efficient communication via selective information revelation is fundamental to various multi agent settings, e.g., (Xuan, Lesser, and Zilberstein 2001; Wu, Zilberstein, and Chen 2011; Unhelkar and Shah 2016; Dughmi and Xu 2016). This work is the first to use information shaping as a one time offline intervention that is performed in order to facilitate goal recognition.

Background: Planning Under Partial Observability

To support agents with partial knowledge, we follow Bonet and Geffner (2011) and consider contingent planning under partial observability, formulated as follows.

Definition 1 *A* **planning under partial observability with deterministic actions (PPO-det) problem** *is a tuple* $P = \langle \mathcal{F}, \mathcal{A}, I, G, \mathcal{O} \rangle$ *where* $\mathcal{F}$ *is a set of fluent symbols,* $\mathcal{A}$ *is a set of actions,* I *is a set of* clauses *over fluent-literals defining the initial situation,* G *is a set of fluent-literals defining the goal condition, and* $\mathcal{O}$ *represents the agent sensor model.*

An action $a \in \mathcal{A}$ is associated with a set of preconditions $prec(a)$, which is the set of fluents that need to hold for a to be applicable, and conditional effects $\mathit{eff}(a)$, which is a set of pairs $(\mathcal{F}_{cond}, \mathcal{F}_{\mathit{eff}})$ s.t. $\mathcal{F}_{\mathit{eff}} \subseteq \mathcal{F}$ become true if $\mathcal{F}_{cond} \subseteq \mathcal{F}$ are true when a is executed. The sensor model $\mathcal{O}$ is a set of observations $o \in \mathcal{O}$ represented as pairs (C, L) where C is a set of fluents and L is a positive fluent, indicating that the value of L is observable when C is true. Each observation $o = (C, L)$ can be conceived as a sensor on the value of L that is activated when C is true.

A state s is a truth valuation over the fluents $\mathcal{F}$ ('true' or 'false'). For an agent, the value of a fluent may be known or unknown. A fluent is *hidden* if its true value is unknown. A *belief state* b is a non-empty collection of states the agent deems possible at some point. A formula $\mathbb{F}$ *holds* in b if it holds for every state $s \in b$. An action a is *applicable* in b if the preconditions of a hold in b, and the *successor* belief state b' is the set of states that results from applying the action a to each state s in b. When an observation $o = (C, L)$ is activated, the successor belief is the *maximal* set of states in b that agree on L. The initial belief is the set of states that satisfy I, and the goal belief are those that satisfy G. A formula is *invariant* if it is true in each possible initial state, and remains true in any state that can be reached from the initial state. A *history* is a sequence of actions and beliefs $h = b_0, a_0, b_1, a_1, \ldots, b_n, a_n, b_{n+1}$. It is *complete* if the performing agent reaches a goal belief state. Finally, a PPO-det problem is *simple* if the non-unary clauses in I are all invariant, and no hidden fluent appears in the body of a conditional effect. We hereon assume our PPO-det problems are simple with connected state spaces.

A solution to a PPO-det problem is a *policy* π, a partial function from beliefs to actions. A policy is *deterministic* if any belief is mapped to at most one action. Otherwise it is *non-deterministic*. A history h *satisfies* π, if $\forall i\ 0 \leq i \leq n$, $a_i \in \pi(b_i)$. There are three policy types: *weak*, when there is at least one complete history that satisfies the policy, *strong*, where a goal belief is guaranteed to be achieved within a fixed number of steps, and *strong cyclic*, where a goal belief is guaranteed to be achieved, but with no upper bound on the cost (length) of the solution (Cimatti et al. 2003). Our framework supports all three policy types.

Goal Recognition Design for Agents with Partial Knowledge (GRD-APK)

The *goal recognition design for agents with partial knowledge problem* (GRD-APK) consists of an initial goal recognition setting, a measure by which a setting is evaluated, and a design model, which specifies the available information shaping modifications.

Goal Recognition

A goal recognition setting can be defined in various ways (Sukthankar et al. 2014), but typically includes a description of the underlying environment, the way agents behave in it to achieve their goal, and the observations collected by the goal recognizing agent.

We have two agents. The *actor* is a partially-informed contingent planner (Definition 1) with a goal, that executes history h until reaching a goal belief or halting when no action is applicable. The *recognizer* is a perfectly informed agent, that analyzes the actor's state transitions in order to recognize the actor's goal as quickly as possible.

Definition 2 *A* ***goal recognition for agents with partial knowledge problem*** *(GR-APK) is a tuple* $R = \langle E, \mathcal{G}, \mathcal{O}^{ac}, \{\Pi(G)\}_{G \in \mathcal{G}} \rangle$ *where:*

- $E = \langle \mathcal{F}, \mathcal{A}, I \rangle$ *is the environment, which consists of the fluents* $\mathcal{F}$*, actions* $\mathcal{A}$ *and initial state* I *as defined in Definition 1 (a cost* $\mathcal{C}(a)$ *for each action* $a \in \mathcal{A}$ *may also be specified),*
- $\mathcal{G}$ *is a set of possible goals* G*, s.t.* $|\mathcal{G}| \geq 2$ *and* $G \subseteq \mathcal{F}$*,*
- $\mathcal{O}^{ac}$ *is the actor's sensor model (Definition 1), and*
- $\{\Pi(G)\}_{G \in \mathcal{G}}$ *are the set of policies* $\Pi(G)$ *an agent aiming at goal* $G \in \mathcal{G}$ *may follow.*

The cost of history h, denoted $\mathcal{C}_a(h) = \Sigma_i \mathcal{C}(a_i)$, is the accumulated cost of the performed actions (equivalent to path length when action cost is uniform). In executing h, the actor follows a possibly non-deterministic policy π from the set $\Pi(G)$ of possible policies to its goal.

The set $\Pi(G)$ of policies to each goal is typically implicitly defined via the solver used by the actor to decide how

to act in each belief state. In Example 1 we described an example of such a solver, which we will formally define in the next section. The GRD-APK framework is well defined for any solver that provides a mapping $\mathcal{B} \rightarrow 2^{\mathcal{A}}$, specifying the set of possible actions an agent may execute at each reachable belief state $b \in \mathcal{B}$, e.g., (Bonet and Geffner 2011; Muise, Belle, and McIlraith 2014).

The actor and recognizer both know the environment E and the set $\mathcal{G}$ of possible goals. While the partially informed actor needs to collect information about the environment via its sensor model $\mathcal{O}^{ac}$ in order to achieve its premeditated goal, the recognizer knows the true state of the world and the actor's solver and sensor, but not its goal. The recognizer observes the actor's transitions between belief states and analyzes them in order to recognize the actor's goal.

Evaluating a GR-APK model

The *worst case distinctiveness* (*WCD*) is the maximum number of actions an actor can perform before its goal is revealed. This maximum depends on the environment, the set of possible goals, the sensor model, and the set of policies to each goal. To define *WCD* we first define the relationship between a history h and a goal. A history *satisfies* a policy if it is a possible execution of the policy.A history *satisfies* a goal if it satisfies a possible policy to the goal.

Definition 3 *Given a GR-APK model R, history h* **satisfies** *policy π in R, if $\forall i\ 0 \leq i \leq n$, $a_i \in \pi(b_i)$. In addition, h* **satisfies** *goal $G \in \mathcal{G}$ in R if $\exists \pi \in \Pi(G)$ s.t. h satisfies π.*

Let $\mathcal{G}^{rec}(h)$ represent the set of goals that history h satisfies, i.e., the set of goals the recognizer deems as possible actor goals. A history is *non-distinctive* if it satisfies more than one goal.

Definition 4 *Given a GR-APK model R, a history h is* **non distinctive** *in R, if exists $G, G' \in \mathcal{G}$ s.t. $G \neq G'$, and h satisfies G and G'. Otherwise, it is distinctive.*

We denote the set of non-distinctive histories of a GR-APK model R by $H_{nd}(R)$.

Definition 5 *The* **worst case distinctiveness** *of a model R, denoted by* $\text{WCD}(R)$ *is:*

$$\text{WCD}(R) = \begin{cases} \max\limits_{h \in H_{nd}(R)} \mathcal{C}_a(h) & H_{nd}(R) \neq \emptyset \\ 0 & otherwise \end{cases}$$

That is, *WCD* is the maximum cost of a history for which the goal is not determined, or zero if there is no such history. Recall that in some instances the goal may remain unrecognized, and even go unattained, in which case *WCD* may be the number of actions (or accumulated action cost) until the end of execution. Also recall that a policy may be strong cyclic, potentially containing infinite loops. A policy with such a cycle is considered to have a history with infinite cost. In particular, since such a history may be non-distinctive, this means *WCD* in this setting may be infinite.

Information Shaping

By using information shaping to change the actor's knowledge, the recognizer can potentially change the actor's behavior in a way that more quickly reveals its goal. We restrict information shaping to be truthful so that it cannot convey false information. In particular, the recognizer can only improve the actor's sensor model, i.e., facilitate its ability to access the value of an environment feature. Let $\boldsymbol{\mathcal{O}}$ denote the set of all sensor models. We define *sensor extension* modifications, which add a single observation to a sensor model.

Definition 6 *A modification $\delta : \boldsymbol{\mathcal{O}} \rightarrow \boldsymbol{\mathcal{O}}$ is a* **sensor extension** *if $\delta(\mathcal{O}) = \mathcal{O} \cup \{o\}$, for all $\mathcal{O} \in \boldsymbol{\mathcal{O}}$ and for some $o = (C, L)$.*

Sensor extensions correspond to adding new sensors to the environment, or, as a special case, communicating to the actor the value of a feature. To demonstrate, in Example 1 the recognizer could allow the actor to sense a stench two (rather than one) cells away from the wumpus in cell (3,B) through a new visual indication. We would add observation $o = (C = AgentAtCell(1, B), L = StenchAtCell(2, B))$ to the actor's sensor model. The recognizer could also directly inform the actor whether there is a wumpus in a specific cell. (e.g., $(C = True, L = WumpusAtCell(3, B))$.

We are now ready to define a GRD-APK problem. Specifically, to support settings where communication may be limited and costly, we use a design budget that limits the number of allowed modifications.

Definition 7 *A* **goal recognition design for agents with partial knowledge problem** *(GRD-APK) is defined as a tuple $T = \langle R_0, \boldsymbol{\Delta}, \beta \rangle$ where:*

- *R_0 is the initial goal recognition model,*
- *$\boldsymbol{\Delta}$ are the possible sensor extensions, and*
- *β is a budget on the number of allowed extensions.*

We want to find a set $\Delta \subseteq \boldsymbol{\Delta}$ of up to β sensor extensions to apply to R_0 to minimize *WCD*:

$$\begin{aligned} &WCD^{min}(T) = \min_{\Delta \subseteq \boldsymbol{\Delta}} \quad WCD(R_0^{\Delta}) \\ &s.t. |\Delta| \leq \beta \end{aligned} \tag{1}$$

Here, $WCD^{min}(T)$ is the minimum *WCD* achievable in a GRD-APK model T, and R^{Δ} is the goal recognition model that results from applying set Δ to R. Any solution to Equation 1 is *optimal*. A solution is *strongly optimal* if it has the minimum number of sensor extensions among all optimal solutions.

The $K_{prudent}(P)$ Translation

A variety of solvers have been developed to solve a PPO-det problem (e.g., (Bonet and Geffner 2011; Muise, Belle, and McIlraith 2014; Brafman and Shani 2012b)), all of which can be used to represent the actor. Specifically, Bonet and Geffner (2011) suggest the *k-planner* that follows the *planning under optimism* approach. Optimism means that the actor plans while making the most optimistic assumptions about missing information, choosing assumptions for

which the corresponding plan has minimal cost. The actor executes the plan from the resulting classical planning problem, revising the assumptions and re-planning if during execution an observation is made that refutes the assumptions.

To transform the PPO-det problem into a classical planning problem, the k-planner uses the $K(P)$ translation. This substitutes each literal 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, and Geffner 2009). Each original action $a \in \mathcal{A}$ is transformed into an equivalent action $a' \in \mathcal{A}'_{exe}$ that replaces the use of every literal L ($\neg L$), with its corresponding fluent KL ($K\neg L$). Each observation $o = (C, L)$ is translated into two deterministic sensing actions $a'_{C,L}, a'_{C,\neg L} \in \mathcal{A}'_{sen}$, one for each possible value of L. These sensing actions allow the solver to compute a plan while choosing to make assumptions about unknown variables. For example, the actor can assume there is no stench in a cell on its plan (e.g., $K\neg StenchAt(1, D) = True$). Each invariant clause is translated into a set of actions $\mathcal{A}'_{ram}$, called the *ramification actions*. These actions can be used to set the truth value of some variable as new sensing information is collected. For example, a ramification action can be activated to deduce that a cell is safe when no breeze or stench is sensed in an adjacent cell. This representation captures the underlying planning problem at the knowledge level, accounting for the exploratory behavior of a partially informed agent.

A key issue to note about the $K(P)$ compilation is that all its actions, including sensing and ramification actions, have equal cost. This means, for example, that a cost-minimizing solution to the resulting classical planning problem may be one that favors increasing the cost to goal over the use of multiple ramification actions. As described in Example 1, we want a solver that can make optimistic assumptions, but that chooses a minimal cost plan that requires making as few assumptions as possible. In addition, ramifications are not to be considered when calculating the cost to goal. For this reason, we introduce the $K_{prudent}(P)$ translation, which extends the uniform cost $K(P)$ translation by associating a cost function to each action. Specifically, every transformed action in $\mathcal{A}'_{exe}$ is assigned a cost of 1, every sensing action (assumption) in $\mathcal{A}'_{sen}$ is assigned a small cost of ϵ, and every ramification action in $\mathcal{A}'_{ram}$ has 0 cost. When ϵ is small enough such that the accumulated cost of assumptions of any generated plan is guaranteed to be smaller than minimal diversion from an optimal plan, the optimal plan achieved using this formulation is a plan with the minimal number assumptions among the set of plans that minimize the accumulated cost of execution actions. [1]

Methods for Information Shaping

To find a set of information shaping modifications that minimize *WCD*, we follow Keren, Gal, and Karpas (2018) and formulate the design process as a search in the space of modification sets $\Delta \subseteq \mathbf{\Delta}$. 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. To calculate *WCD*, we use the actor's solver to compute the policies it may follow to each goal, and find the maximal cost of a non-distinctive policy prefix (Definition 4), i.e., a prefix shared by policies to more than one goal.

Given a goal recognition model R and a sensor extension δ, we let R^{δ} denote the model that results from applying δ to the actor's sensor model $\mathcal{O}$. A sensor extension is useful with regards to a goal recognition model if it reduces *WCD*.

Definition 8 *A sensor extension δ is* **useful** *with regards to goal recognition model R if* $\text{WCD}(R^{\delta}) < \text{WCD}(R)$.

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 (i.e., computing *WCD* of the modified model) may be costly, making it important to develop efficient ways to search the space of design options. Second, the problem is non-monotonic, in that sensor extensions are not always useful, and providing more information can both increase or decrease *WCD* (Example 1). This affects the methods we can use for reducing the state space while guaranteeing completeness. Specifically, we cannot use techniques suggested for previous GRD models, such as those suggested by Keren, Gal, and Karpas (2018), which rely in the assumption that modifications cannot increase *WCD*.

Design with CG-Pruning

The baseline approach for searching in the modification space is *breadth first search* (BFS), using *WCD* 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 if a model with $WCD = 0$ is found or if there are no more nodes to explore, and returns the shortest path (smallest modification set) to a node that achieves minimal *WCD*. This is guaranteed to find a strongly optimal solution, but does not scale to larger problems.

To improve computational efficiency, we suggest a new pruning approach that reduces our search space by eliminating useless modifications. We then specify conditions under which our pruning is *safe* (Wehrle and Helmert 2014), s.t. at least one optimal solution remains unpruned

The high level idea of our pruning technique, dubbed CG-Pruning, 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 affect the actor's behavior and are therefore not useful.

Specifically, given a goal recognition 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 and Nayak 1997; Helmert 2006) of each transformed problem. The causal graph of a planning problem is a directed graph (V, E) where the nodes V represent the state variables and

[1] The formal definitions and proofs for this and next sections are given in the attached appendix.

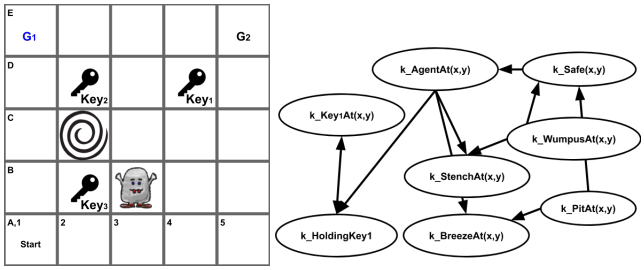


Figure 2: The Wumpus domain with keys

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.

In our context, the variable set of the causal graph can either be the set of fluents of the transformed PPO-det problem, or the multi-valued variables extracted using *invariant synthesis*, which automatically finds sets of fluents among which exactly one is true at each state (e.g., in Example 1 the fluents representing whether an agent is in a specific location can be aggregated into a single multi-valued variable representing the agent's location). In any case, the casual graph of the transformed planning problem to each possible goal captures all variables relevant for achieving the goal and the hierarchical dependencies between them.

Recall that each sensor extension is characterized by an observation $o = (C, L)$ that is added to the actor's sensor model. CG-Pruning prunes all sensor extensions for which the fluents corresponding to knowledge about L in the transformed problem (i.e. KL and $K\neg L$) do not occur in any of the casual graphs to the goals.

To demonstrate, consider a variation of Example 1 depicted in Figure 2(left), where keys are distributed in initially unknown locations on the grid, and the actor needs to collect the key to its goal in order to access it (K_HoldingKey$_i$ is a precondition to accessing goal G_i). 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. Figure 2 (right), shows a part of the causal graph for G_1, excluding variables concerning the irrelevant keys. By generating the causal graph to all goals, CG-Pruning automatically detects and prunes sensor extensions related to Key_3.

Theoretical Analysis

In order to safely prune sensor extensions from our search space, we use the causal graph of the actor's planning problem. Since the causal graph encapsulates any information that may affect the actor's behavior, we can use it to automatically detect non-useful sensor extensions. Recall that our actor uses the *k-planner* to iteratively compute a plan at the initial state, and every time an assumption made at a previous iteration is refuted. At each iteration, the actor's current partially observable problem is transformed (using either $K(P)$ or $K_{prudent}(P)$) into its corresponding fully observable problem, and the new plan is the solution of the transformed problem given by an off-the-shelf classical planner. For each model R and execution iteration i, we let $CG_i^R(G)$ represent the causal graph of the transformed planning problem to goal G at iteration i. CG-Pruning prunes sensor extensions that reveal information about variables that are not in $CG_0^R(G)$ for any goal $G \in \mathcal{G}$.

Theorem 1 *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 solver.*

Proof:(sketch) For any goal $G \in \mathcal{G}$, the causal graph $CG_i^R(G)$ at each iteration $0 \leq i$ is a supergraph of any causal graph $CG_j^R(G)$ of a subsequent iteration for all $i \leq j$. The only difference between the compiled planning problem at each iteration is the initial belief state, which does not affect the causal graph.

When an optimal solver is used for the transformed problem, then a sensor extension δ that adds observation $o = (C, L)$ s.t. neither KL nor $K\neg L$ appear in $CG_0^R(G)$ for any of the goals $G \in \mathcal{G}$, is not useful. This is based on Bonet and Geffner (2011)'s proof that the $K(P)$ (and $K_{prudent}(P)$) transformation is sound and complete for simple problems with a connected space, which are the only problems we consider here. In addition, an optimal plan of a classical planning problem will not include actions that do not affect variables in its causal graph (Helmert 2006). Therefore, for any design node Δ, only sensor extensions that affect a variable in a causal graph to some goal can possibly affect the actor's behavior. Other extensions will leave the *WCD* unchanged and are not useful. Therefore CG-Pruning is safe. ■

Empirical Evaluation

In this section, we report experiments that demonstrate both the effect of sensor extensions on *WCD* as well as the computational advantage provided through CG-Pruning.

Dataset. We use seven domains, adapted from Bonet and Geffner (2011) and Albore, Palacios, and Geffner (2009). The adaptation from partially observable planning to GRD-APK involves specifying for each instance the set of possible goals and sensor extensions (see Table 1 for details).

To support the design process, we use STRIPS (Fikes and Nilsson 1972) to specify the available modifications and their effect. Sensor extensions are implemented as design actions that add to the initial state fluents that indicate the true value of a variable.

Setup. We use the *k-planner* (Bonet and Geffner 2011) as the actor's solver, with two variations: (1) the $K(P)$ compilation is used together with the satisfying FF classical planner (FF) (Hoffmann and Nebel 2001) (2) the $K_{prudent}(P)$ compilation is used together with the optimal Fast-Downward (Helmert 2006) classical planner (FD), using the lm-cut heuristic (Helmert and Domshlak 2009).

The design process is implemented as a breadth-first search (BFS) in the space of modification sets, tested with and without CG-Pruning. To parse the design file, we adapt *pyperplan* (Alkhazraji et al. 2016) to provide for each modification set (representing a GRD-APK model and a search

Domain	Description	Possible Goals	Sensor Extensions
WUMPUS	the setting described in Example 1	gold locations	safe cells
WUMPUS-KEY	the setting depicted in Figure 2	gold locations	safe cells or locations with / without keys
COLOR-BALLS	the actor navigates a grid to deliver balls of different and initially unknown colors to their per-color destinations	ball distribution	locations with / without a ball
TRAIL	the actor must follow a trail to reach a destination, while sensing the reachable cells surrounding it	final stone locations	locations with / without a stone
LOGISTICS	trucks transport packages to their destinations, relying on sensing to reveal the packages in a location	package destination	package locations
LOGISTICS-I	same as LOGISTICS but without additional information (no initial information about package locations)	as for LOGISTICS	as for LOGISTICS
UNIX	a user wants to transfer a file without knowing its initial location in a folder tree	file destinations	locations with/ without a file

Table 1: Domain description, possible goals and design options for each domain.

	No Pruning				CG-Pruning			
	sol	ΔWCD	time	nodes	sol	ΔWCD	time	nodes
WUMPUS	0.10	0.28 (0.32)	1184.32	25.11	0.21	0.28 (0.32)	854.62	22.51
WUMPUS-KEY	0.13	0.09 (0.10)	852.32	14.87	0.26	0.09 (0.10)	642.53	5.71
COLOR-BALLS	0.66	0.11(0.33)	801.07	15.2	0.67	0.11 (0.33)	637.72	12.3
TRAIL	0.17	0.13 (0.09)	593.97	85.2	0.19	0.13 (0.09)	484.65	76.36
LOGISTICS	0.5	2.22(0.81)	824.35	15.5	0.7	2.22 (0.81)	740.24	10.3
LOGISTICS-I	0.4	3.91 (0.32)	18.3	562.4	0.7	3.91 (0.32)	15.1	432.5
UNIX	0.9	2.62 (0.59)	179.6	79.0	1.0	2.62 (0.59)	49.2	22.3

Table 2: Results per domain for the optimal FD solver

	No Pruning				CG-Pruning			
	sol	ΔWCD	time	nodes	sol	ΔWCD	time	nodes
WUMPUS	0.84	2.12 (0.41)	233.58	45.74	0.84	2.12 (0.41)	148.57	31.23
WUMPUS-KEY	0.69	3.38 (0.77)	23.86	28.84	0.73	3.27(0.75)	7.29	14.15
COLOR-BALLS	1.0	2.06 (0.29)	29.32	5.75	1.0	2.06 (0.29)	55.23	5.75
TRAIL	1.0	0.10 (0.12)	70.65	127.25	1.0	0.10 (0.12)	107.66	110.09
LOGISTICS	0.95	1.40 (0.57)	54.55	17.78	1.0	1.40 (0.57)	81.47	3.43
LOGISTICS-I	0.43	0.26 (0.11)	53.5	93.5	0.92	0.26 (0.11)	61.5	24.2
UNIX	1.0	2.67 (0.51)	125.2	283.3	1.0	2.67 (0.51)	40.3	83.3

Table 3: Results per domain for the non-optimal FF solver

node) the set of applicable modifications and resulting models. We use 40 instances for each domain, using design budgets of 1 and 2. We fix the time limit for an execution to 20 minutes and 1000 search steps (each corresponding to a modification set), whichever was first.

Results. Tables 2 and 3 summarize the results for the FD and FF solvers, respectively. For each domain the tables show 'sol' as the fraction of instances completed within the time and resource bounds. For instances completed by both approaches 'ΔWCD' is the average *WCD* reduction achieved via design, i.e., the *WCD* difference between the original setting and one where sensor extensions are applied (normalized values are given in parentheses). Recall that *WCD* also accounts for prefixes of failed executions, since they represent valid agent behavior. For instances completed by both approaches, we give the average calculation time (in seconds) and average number of nodes evaluated.

The results show that design via information shaping is effective in reducing *WCD* across each of our domains. In addition, by automatically detecting and excluding futile sensor extensions, CG-Pruning increases the ratio of problems for which computation completed without reaching the time or resource bounds in the WUMPUS, WUMPUS-KEY, LOGISTICS, and LOGISTICS-I domains. CG-Pruning also reduces the number of nodes explored and computation time for problems completed by both approaches for all domains except C-BALLS and TRAIL with FF.

Discussion. It is the trade off between the costs of the causal graph extraction and node computation that dictates the benefit of using CG-Pruning. Specifically, our results demonstrate that our pruning is effective when the problem description includes many variables that do not affect the actor's behavior (e.g., packages that don't appear in the goal in the LOGISTICS-I domain). Understanding which information is irrelevant to the goal is not always a trivial task, especially since our models include invariant information, creating additional relationships between variables. The key benefit of the causal graph analysis is in automatically detecting and excluding futile sensor extensions. Our results highlight the domains for which the overhead of the causal graph analysis pays-off, especially as the problem size increases. For example, in LOGISTICS-I with FF for smaller problems completed by both approaches, the reduction in the number of computed nodes is overcast by the higher computation time required by CG-Pruning. However, CG-Pruning solves more than twice as much instances without exhausting the resource bounds.

Note that although our approach is provably safe only when the actor uses an optimal planner (e.g. FD), the computational benefits of our approach are apparent also when the sub-optimal FF planner is used. In most cases, CG-Pruning achieves the same *WCD* reduction as the exhaustive search, while saving on computation time (e.g. 70% saving for LOGISTICS-I), and solving more instances.

Conclusion

We introduced GRD for a partially informed actor and a perfectly informed recognizer, which is able to share information with the actor about the domain. We formalized the information shaping problem as one of minimizing worst-case distinctiveness (*WCD*), using sensor extensions to improve goal recognition. We presented a new solver for planning under partial observability, and studied the use of a safe pruning method together with breadth first search to search the space of applicable sensor extensions, introducing the idea of using techniques developed for classical planning for supporting the goal recognition of partially informed planning agents. Our results on standard benchmark domains show that WCD can be reduced via information shaping and demonstrate the effectiveness of pruning.

This work provides a first step in supporting GRD for partially informed agents. In future work, it will be interesting to use quantitative models, and Partially Observable Markov Descision Process (POMDP) models (Kaelbling, Littman, and Cassandra 1998) in particular, to represent the actor and its belief state. Future work should also consider settings where the actor is aware of the recognizer's presence, and where information shaping can be applied online, based on the actor's actual progress.

References

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In *IJCAI*.

Alkhazraji, Y.; Frorath, M.; Grutzner, M.; Liebetraut, T.; Ortlieb, M.; Seipp, J.; Springenberg, T.; Stahl, P.; Wulfing, J.; Helmert, M.; and Mattmuller, R. 2016. Pyperplan: https://bitbucket.org/malte/pyperplan.

Ang, S.; Chan, H.; Jiang, A. X.; and Yeoh, W. 2017. Game-theoretic goal recognition models with applications to security domains. In *International Conference on Decision and Game Theory for Security*.

Boddy, M. S.; Gohde, J.; Haigh, T.; and Harp, S. A. 2005. Course of action generation for cyber security using classical planning. In *ICAPS*.

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

Brafman, R., and Shani, G. 2012a. A multi-path compilation approach to contingent planning. In *AAAI*.

Brafman, R., and Shani, G. 2012b. Replanning in domains with partial information and sensing actions. *Journal of Artificial Intelligence Research (JAIR)* 45.

Carberry, S. 2001. Techniques for plan recognition. *User Modeling and User-Adapted Interaction* 11.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence 2003* 147.

Cohen, P. R.; Perrault, C. R.; and Allen, J. F. 1981. Beyond question-answering. Technical report, DTIC Document.

Dughmi, S., and Xu, H. 2016. Algorithmic bayesian persuasion. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*.

Fikes, R. E., and Nilsson, N. J. 1972. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2(3):189–208.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *ICAPS*.

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

Hoffmann, J., and Nebel, B. 2001. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research (JAIR)* 14.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence* 101(1-2).

Kautz, H., and Allen, J. F. 1986. Generalized plan recognition. In *AAAI*, volume 86.

Kautz, H.; Etzioni, O.; Fox, D.; Weld, D.; and Shastri, L. 2003. Foundations of assisted cognition systems. Technical report, University of Washington.

Keren, S.; Gal, A.; and Karpas, E. 2014. Goal recognition design. In *ICAPS*.

Keren, S.; Gal, A.; and Karpas, E. 2015. Goal recognition design for non optimal agents. In *AAAI*.

Keren, S.; Gal, A.; and Karpas, E. 2016a. Goal recognition design with non-observable actions. In *AAAI*.

Keren, S.; Gal, A.; and Karpas, E. 2016b. Privacy preserving plans in partially observable environments. In *IJCAI*.

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

Levine, S. J., and Williams, B. C. 2014. Concurrent plan recognition and execution for human-robot teams. In *ICAPS*.

Muise, C. J.; Belle, V.; and McIlraith, S. A. 2014. Computing contingent plans via fully observable non-deterministic planning. In *AAAI*.

Ramirez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *AAAI*.

Russell, S. J., and Norvig, P. 2016. *Artificial intelligence: a modern approach*. Pearson Education.

Sukthankar, G.; Geib, C.; Bui, H. H.; Pynadath, D.; and Goldman, R. P. 2014. *Plan, activity, and Intent Recognition: Theory and practice*. Newnes.

Unhelkar, V. V., and Shah, J. A. 2016. Contact: Deciding to communicate during time-critical collaborative tasks in unknown, deterministic domains. In *AAAI*.

Wayllace, C.; Hou, P.; Yeoh, W.; and Son, T. C. 2016. Goal recognition design with stochastic agent action outcomes". In *IJCAI*.

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

Williams, B. C., and Nayak, P. P. 1997. A reactive planner for a model-based executive. In *IJCAI*, volume 97.

Wu, F.; Zilberstein, S.; and Chen, X. 2011. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence* 175(2).

Xuan, P.; Lesser, V.; and Zilberstein, S. 2001. Communication decisions in multi-agent cooperation: Model and experiments. In *Proceedings of the fifth international conference on Autonomous agents*.

Zhang, H.; Chen, Y.; and Parkes, D. C. 2009. A general approach to environment design with one agent. In *IJCAI*.