Accounting for Partial Observability in Stochastic Goal Recognition Design: Messing with the Marauder's Map

Christabel Wayllace[†] and **Sarah Keren**[‡] and **William Yeoh**[†]
and **Avigdor Gal**[‡] and **Erez Karpas**[‡]
[†]Washington University in St. Louis
[‡]Technion – Israel Institute of Technology

Abstract

Given a stochastic environment and a set of allowed modifications, the task of goal recognition design is to select a valid set of modifications that minimizes the expected maximal number of steps an agent can take before his goal is revealed to an observer. This paper extends the *stochastic goal recognition design* (S-GRD) framework in the following two ways: (1) Agent actions are unobservable; and (2) Agent states are only partially observable. These generalizations are motivated by practical applications such as agent navigation, where agent actions are unobservable yet his state (current location) can be (at least partially) observed, using possibly low sensor (e.g., GPS) resolution, forcing nearby states to become indistinguishable. In addition to the generalized model, we also provide accompanying algorithms that calculate the expected maximal number of steps, offer new sensor refinement modifications that can be applied to enhance goal recognition, and evaluate them on a range of benchmark applications.

1 Introduction

Goal recognition aims at discovering the goals of an agent according to his observed behavior, collected online [Carberry, 2001; Ramírez and Geffner, 2010; Sukthankar *et al.*, 2014]. *Goal recognition design* (GRD) [Keren *et al.*, 2014] is the offline task of redesigning environments (either physical or virtual) to allow efficient online goal recognition.

Typically, a GRD problem has two components: (1) The goal recognition setting being analyzed and a measure of the efficacy of goal recognition and (2) a model of possible design changes one can make to the underlying environment. In the seminal work by Keren *et al.* [2014], they proposed the *worst case distinctiveness* (*wcd*) metric, which aims at capturing the maximum number of steps an agent can take without revealing its goal, as a measure of the goal recognition efficacy. Removal of actions was considered as a possible design change to the environment. This definition is made for the problem under three key assumptions:

- **Assumption 1**: Agents in the system execute optimal plans to reach their goals;
- **Assumption 2**: The environment is fully observable (i.e., both states and actions of agents are observable); and
- **Assumption 3**: Agent actions are deterministic.

The GRD problem has been since generalized to relax each of the three assumptions [Keren *et al.*, 2015; 2016a; 2016b; Wayllace *et al.*, 2016; 2017; Keren *et al.*, 2018]. Aside from these relaxations, Wayllace *et al.* [2017] have also proposed a new metric called *expected case distinctiveness* (*ecd*), which weighs the possible goals based on their likelihood of being the true goal. Additionally, Keren *et al.* [2016b] have proposed the refinement of sensors, which decreases the degree of observation uncertainty on tokens produced by actions (rather than states) of an agent, as a possible design change to the environment. Table 1 summarizes the generalizations, metrics, and possible designs of existing GRD models.

In this work, we go beyond the state-of-the-art by extending the *Stochastic GRD* (S-GRD) model [Wayllace *et al.*, 2017] to also relax Assumption 2 and handle partially observable environments. The new model, which we call *Partially-Observable Stochastic GRD* (POS-GRD), assumes that actions of the agent are now no longer observable and states of the agent are now partially observable. This relaxation is motivated by practical applications such as agent navigation, where agent actions are unobservable yet his state (current location) can be (at least partially) observed. The partial observability of agent states is due to low sensor (e.g., GPS) resolution – several nearby states may be indistinguishable from one another. Finally, we also consider sensor refinement as a possible design to the environment, which contributes to the observability of states.

Our empirical evaluation shows that partial observability increases the *wcd* required to recognize the agent's goal and that sensor refinement always reduces this value. The analysis also suggests that, given a limited number of possible modifications, the initial sensor configuration affects the value of *wcd* and its reduction ratio; therefore, it might be possible to reduce the *wcd* even

	Generalizations			Metrics		Possible Designs	
	Suboptimal Plans	Partially Obs. Env.	Stochastic Actions	*wcd*	*ecd*	Action Removal	Sensor Refinement
Keren *et al.* [2014]				✓		✓	
Son *et al.* [2016]				✓		✓	
Keren *et al.* [2015]	✓			✓		✓	
Keren *et al.* [2016a]	✓	✓		✓		✓	
Keren *et al.* [2016b]	✓	✓		✓		✓	✓
Wayllace *et al.* [2016]			✓	✓		✓	
Wayllace *et al.* [2017]			✓	✓	✓	✓	
Our proposed model		✓	✓	✓		✓	✓

Table 1: Properties of Current Goal Recognition Design Models

further using the same number of sensors with the same resolution but in a different configuration.

2 Illustrating Example

To illustrate the setting of this work, we present an example from the wizarding world of Harry Potter, who is back at the Hogwarts School of Witchcraft and Wizardry. He is tasked with establishing a security system that can detect as early as possible a student who enters the school from the main entrance and is heading towards Professor Snape's office armed with a wand made of oak wood. All students use the staircase chamber to move around the school. The staircase is stochastic, especially when a student is walking up the staircase. Therefore, a student aiming at a certain part of the school may find herself at a different location. For example, a student heading to Professor Snape's office or the dining hall may find herself at the hallway, in which case she needs to retrace to the entrance and try reaching her destination again. Figure 1(top) depicts the locations as nodes and the transitions (and their probabilities) as edges.

To accomplish his task, Potter plans to use the Marauder's Map, a magical artifact that reveals the whereabout of all witches and wizards at Hogwarts. The map can show where witches and wizards are, but due to some dark magic, it can no longer identify them by their names. Further, it was not created with the ability to detect whether a student carries a wand, not to mention the type of wood of which the wand is made.

Potter can cast exactly *one* spell to either reveal the name of the witches and wizards on the map or to reveal the type of wand that they are carrying, if any. Knowing that all wands are forbidden in the dining hall, Potter realizes that his best choice is to cast the spell that reveals wands and the type of wood of which they are made. This will guarantee that anyone ending up at the hallway with a wand made of oak wood and heads back to the entrance has the intention of reaching Professor Snape's office, and such a recognition can occur after at most two actions, namely moving towards Professor Snape's office but ending up in the hallway and returning to the entrance. Figure 1(bottom) illustrates

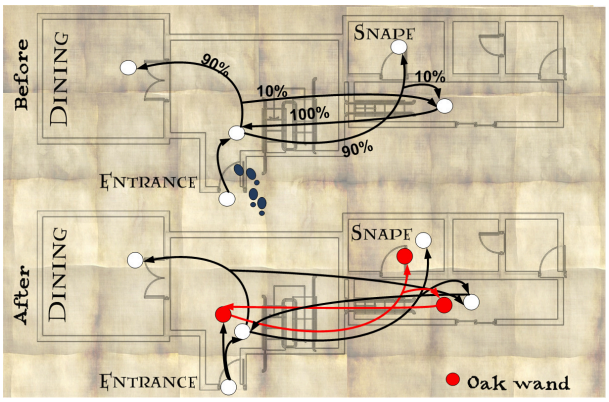


Figure 1: Marauder's Map Before (top) and After (bottom) Potter's Modification Spell

the problem after the modification spell; each node now represents a ⟨ location, wand ⟩ tuple, where nodes in red represent the states with oak wands.

3 Background

3.1 Markov Decision Process (MDP)

A *Stochastic Shortest Path Markov Decision Process* (SSP-MDP) [Mausam and Kolobov, 2012] is represented as a tuple $\langle \mathbf{S}, s_0, \mathbf{A}, \mathbf{T}, \mathbf{C}, \mathbf{G} \rangle$. It consists of a set of states $\mathbf{S}$; a start state $s_0 \in \mathbf{S}$; a set of actions $\mathbf{A}$; a transition function $\mathbf{T} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \to [0, 1]$ that gives the probability $T(s, a, s')$ of transitioning from state s to s' when action a is executed; a cost function $\mathbf{C} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \to \mathbb{R}$ that gives the cost $C(s, a, s')$ of executing action a in state s and arriving in state s'; and a set of goal states $\mathbf{G} \subseteq \mathbf{S}$. The goal states are terminal, that is, $T(g, a, g) = 1$ and $C(g, a, g) = 0$ for all goal states $g \in \mathbf{G}$ and actions $a \in \mathbf{A}$.

An SSP-MDP must also satisfy the following two conditions: (1) There must exist a *proper policy*, which is a mapping from states to actions with which an agent can reach a goal state from any state with probability 1. (2) Every *improper policy* must incur an accumulated cost of ∞ from all states from which it cannot reach the goal with probability 1. In this paper, we will focus on SSP-MDPs and will thus use the term MDPs to refer to

SSP-MDPs. A *solution* to an MDP is a policy π, which maps states to actions. Solving an MDP means finding an optimal policy, that is, a policy with the smallest expected cost. Finally, we use *optimal actions* to denote actions in an optimal policy.

3.2 Value Iteration (VI) and Topological VI (TVI)

Value Iteration (VI) [Bellman, 1957] is one of the fundamental algorithms to find an optimal policy. It uses a value function V to represent expected costs. The expected cost of an optimal policy π^* for the starting state $s_0 \in \mathbf{S}$ is the expected cost $V(s_0)$, and the expected cost $V(s)$ for all states $s \in \mathbf{S}$ is calculated using the Bellman equation [Bellman, 1957]:

$$V(s) = \min_{a \in \mathbf{A}} \sum_{s' \in \mathbf{S}} T(s, a, s') \big[C(s, a, s') + V(s') \big] \qquad (1)$$

The action chosen by the policy for each state s is then the one that minimizes $V(s)$.

VI suffers from a limitation that it updates each state in every iteration even if the expected cost of some states have converged. *Topological VI* (TVI) [Dai *et al.*, 2011] addresses this limitation by repeatedly updating the states in only one *strongly connected component* (SCC) until their values converge before updating the states in another SCC. Since the SCCs form a directed acyclic graph, states in an SCC only affect the states in upstream SCCs. Thus, by choosing the SCCs in reverse topological sort order, it no longer needs to consider SCCs whose states have converged in a previous iteration.

3.3 Goal Recognition Design (GRD)

A *Goal Recognition Design* (GRD) problem [Keren *et al.*, 2014] is represented as a tuple $T = \langle P, \mathcal{D} \rangle$, where P is an initial goal recognition model and $\mathcal{D}$ is a design model. The initial model P, in turn, is represented by the tuple $\langle D, G \rangle$, where D captures the domain information and $\mathbf{G}$ is a set of possible goal states of the agent. The *worst case distinctiveness* (*wcd*) of problem P is the length of a longest sequence of actions $\pi = \langle a_1, \ldots, a_k \rangle$ that is the prefix in *cost-minimal* plans $\pi^*_{g_1}$ and $\pi^*_{g_2}$ to distinct goals $g_1, g_2 \in \mathbf{G}$. Intuitively, as long as the agent executes π, he does not reveal his goal to be either g_1 or g_2.

A design model $\mathcal{D}$ [Keren *et al.*, 2018] includes three components: The set $\mathcal{M}$ of modifications that can be applied to a model; a modification function δ that specify the effect each modification $m \in \mathcal{M}$ has on the goal recognition setting to which it is applied; and a constraint function ϕ that specifies the modification sequences that can be applied to a goal recognition model. In the original GRD problem definition, action removals are the only modifications allowed in the design model.

The objective in GRD is to find a feasible modification sequence that, when applied to the initial goal recognition model P, will minimize the *wcd* of the problem. This optimization problem is subject to the requirement that the minimal cost to achieve each goal $g \in \mathbf{G}$ is the same before and after the modifications.

Researchers have proposed a number of extensions to support different goal recognition and goal recognition design models, tabulated in Table 1.

3.4 Stochastic GRD (S-GRD)

Stochastic Goal Recognition Design (S-GRD) [Wayllace *et al.*, 2016; 2017] extends the GRD framework by assuming the actions executed by the agent, which are fully observable, have stochastic outcomes. Similar to GRD, it is represented as a tuple $T = \langle P, \mathcal{D} \rangle$, where $P = \langle D, G \rangle$ is an initial goal recognition model, $\mathcal{D}$ is a design model, D captures the domain information, and $\mathbf{G}$ is a set of possible goal states of the agent.

The elements of $D = \langle \mathbf{S}, s_0, \mathbf{A}, \mathbf{T}, \mathbf{C} \rangle$ of S-GRD problems are as described in MDPs, except that the cost function $\mathbf{C}$ is restricted to positive costs. It is assumed that the cost of all actions is 1 for simplicity. The *worst case distinctiveness* (*wcd*) of problem P is the largest *expected* cost incurred by the agent without revealing his true goal. The *wcd* of a problem assumes that all goals are of equal likelihood of being the true goal. The *expected case distinctiveness* (*ecd*) weighs the expected cost of each policy for a goal by the likelihood of that goal to be the true goal.

Augmented MDP for S-GRDs: Given a regular MDP $\langle \mathbf{S}, s_0, \mathbf{A}, \mathbf{T}, \mathbf{C}, \mathbf{G} \rangle$, an augmented MDP $\langle \tilde{\mathbf{S}}, \tilde{s}_0, \tilde{\mathbf{A}}, \tilde{\mathbf{T}}, \tilde{\mathbf{C}}, \tilde{\mathbf{G}} \rangle$ augments each component of the tuple in the following way:

- Each state $\tilde{s} \in \tilde{\mathbf{S}}$ is represented by $\langle s, \mathbf{G}' \rangle$ where $s \in \mathbf{S}$ and $\mathbf{G}' \subseteq \mathbf{G}$ is the set of possible goals for s. Two augmented states are different if any of their components are different.
- The augmented start state is $\tilde{s}_0 = \langle s_0, \mathbf{G} \rangle$.
- Each augmented action $\tilde{a} \in \tilde{\mathbf{A}}$ is a tuple $\langle a, \mathbf{G}' \rangle$, where $a \in \mathbf{A}$ and $\mathbf{G}'$ is the set of all goals for which that action is an *optimal action*.
- The new transition function $\tilde{\mathbf{T}} : \tilde{\mathbf{S}} \times \tilde{\mathbf{A}} \times \tilde{\mathbf{S}} \to [0, 1]$ gives the probability $\tilde{T}(\tilde{s}, \tilde{a}, \tilde{s}')$, where $\tilde{s} = \langle s, \mathbf{G}' \rangle$, $\tilde{a} = \langle a, \mathbf{G}'' \rangle$, and $\tilde{s}' = \langle s', \mathbf{G}' \cap \mathbf{G}'' \rangle$. $\tilde{T}(\tilde{s}, \tilde{a}, \tilde{s}') = T(s, a, s')$ if $|\mathbf{G}' \cap \mathbf{G}''| > 1$ and equals 0 otherwise.
- The cost function $\tilde{C} : \tilde{S} \times \tilde{A} \times \tilde{S} \to \mathbb{R}^+$ gives the cost $\tilde{C}(\tilde{s}, \tilde{a}, \tilde{s}')$ of executing action $\tilde{a}$ in augmented state $\tilde{s}$ and arriving in $\tilde{s}'$. This cost equals the cost $\tilde{C}(\tilde{s}, \tilde{a}, \tilde{s}') = C(s, a, s')$ under the same conditions as above.
- The augmented goal states $\tilde{\mathbf{G}} \subseteq \tilde{\mathbf{S}}$ are those augmented states $\langle s, \mathbf{G}' \rangle$ for which any execution of an augmented action will transition to an augmented state $\langle s', \mathbf{G}''' \rangle$ with one goal or no goals (*i.e.*, $|\mathbf{G}'''| \leq 1$) in the regular MDP.

S-GRD algorithms use augmented MDPs and VI-like algorithms to compute the *wcd* by finding the maximum expected cost from the augmented starting state to any augmented goal.

4 Partially-Observable S-GRD (POS-GRD)

A key assumption in S-GRDs is that the actions of the agents are observable. However, in applications such as those involving agent navigation, agent actions are not observed and only his state (current location) is available. Further, while in deterministic settings (Assumption 3), one can accurately infer the action of an agent by continuously observing his state, this does no longer hold in S-GRDs. Therefore, the S-GRD algorithms proposed thus far cannot be used off-the-shelf directly.

Towards this end, we define the *Partially-Observable S-GRD* (POS-GRD) problem, where (1) only states (rather than actions) can be observed; and (2) states are partially observable, so that several states are indistinguishable from one another. The degree of observation uncertainty is defined by the resolution of sensors in the problem.

We follow Keren *et al.* [2018] by modeling a POS-GRD problem with two components: A *goal recognition model* that describes the goal recognition problem and a *design model* that specifies the possible ways one can modify the goal recognition model. We formulate each of these components separately before integrating them into a POS-GRD model.

Definition 1 (Goal Recognition Model) *A partially-observable goal recognition (POS-GRD) model with stochastic action outcomes is represented as a tuple* $PO = \langle D, \mathbf{G}, N \rangle$ *where*

- $D = \langle \mathbf{S}, s_0, \mathbf{A}, \mathbf{T}, \mathbf{C} \rangle$ *captures the domain information;*
- $\mathbf{G}$ *is a set of possible goals; and*
- N *represents a sensor function that partitions* $\mathbf{S}$ *into observation sets* $\mathbf{O}_1^N, ..., \mathbf{O}_n^N$*, where* $\forall s_i, s_j \in \mathbf{S}, s_i \neq s_j : s_i, s_j \in \mathbf{O}_k^N \iff N(s_i) = N(s_j)$*. Each set* $\mathbf{O}_k^N$ *corresponds to a different observation and we refer to* $N(s)$ *as the projected observation of* s*.*

The above model generalizes the stochastic goal recognition setting proposed by Wayllace *et al.* [2016] by including a sensor model N that defines the degree of partial observability of the states in the problem. If all the states are fully observable, then observation set $\mathbf{O}_i$ is composed of *exactly* one state.

In this paper, we focus on two types of possible modifications in the design model $\mathcal{D}$:

- Action Removal: A modification that removes some actions from the set of applicable actions in the model, and
- Sensor Refinement: A modification that allows the observer to distinguish between two states that have the same observation.

Definition 2 *[refinement] A sensor model* N' *is a* refinement *of sensor model* N *if there exists a set* $\mathbf{O}_j^N$ *such that* $\mathbf{O}_i^{N'} \subseteq \mathbf{O}_j^N$ *for each observation* $\mathbf{O}_i^{N'}$ *of* N'*.*

We let PO^m represent the POS-GRD model that results from applying m to PO and let N^m and N denote the sensor models of PO^m and PO, respectively. We define sensor refinement as follows.

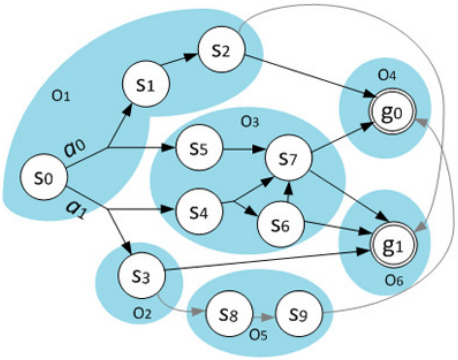


Figure 2: Example Illustration

Definition 3 *[sensor refinement] A modification* m *is a* sensor refinement *modification if for every goal recognition model* PO*,* PO^m *is identical to* PO *except that* N^m *is a refinement of* N*.*

Note that as opposed to the sensor refinement suggested by Keren *et al.* [2016b], where the sensor model is defined over tokens emitted by performed actions, sensor refinement defined here applies to settings where the state of the agent may be only partially observed and the observer has a way to improve its observability by sensing features of the environment.

Definition 4 *A partially-observable goal recognition design (POS-GRD) problem is given by the pair* $T = \langle PO_0, \mathcal{D} \rangle$*, where*

- PO_0 *is an initial goal recognition model, and*
- $\mathcal{D}$ *is a* design model.

Due to low sensor resolution, more than one state can map to the same observation. We represent this fact by grouping states with the same observation together. Figure 2 illustrates the states and their observations, where each shaded area represents one observation. If the agent moves between states with the same observation, it is impossible for the observer to know whether the agent stayed in its current state or moved to a different state with the same observation. If the agent moves to a state that has a different observation, the observer may gain some information. To do so, the observer needs to keep track of the set of possible goals given the sequence of observations observed. Wayllace *et al.* [2017] used an augmented MDP to keep track of the history of states of the agent and to discard some possible goals along the path. To solve a POS-GRD problem, we make use of a similar augmented MDP structure, where we modify it to account for partial observability. Below, we provide new definitions that will be useful to explain the construction of the new augmented MDP.

Definition 5 (Starting State) *State* s *is a starting state if* $\exists s' \in \mathbf{S}, a \in \mathbf{A} : \mathbf{T}\langle s', a, s \rangle > 0 \wedge N(s) \neq N(s') \vee s = s_0$*, where* s_0 *is the initial state.*

Definition 6 (Set of Connected States) *Given a starting state s_i, the set of connected states $\mathbf{C}_i$ of s_i is the set of all states that are reachable from s_i and whose observation is the same as s_i. More precisely,* $\mathbf{C}_i = \{s_i\} \cup \{s_j \mid \mathbf{T}(s_i, \cdot, s_j) > 0 \wedge N(s_j) = N(S_i)\}$.

Definition 7 (Connected Observation) *Given a set of starting states $\mathbf{S}'$, its connected observation is $\tilde{\mathbf{O}}(\mathbf{S}') = \cup_{s_i \in \mathbf{S}'} \mathbf{C}_i$, where $\mathbf{C}_i$ is the set of connected states of starting state s_i.*

Definition 8 (Ending State) *Given a connected observation $\mathbf{O}(\mathbf{S}')$, state s is an ending state of $\mathbf{O}(\mathbf{S}')$ if* $s \in \mathbf{O}(\mathbf{S}') \wedge \exists a \in \mathbf{A}, s' \notin \mathbf{O}(\mathbf{S}') : \mathbf{T}\langle s, a, s'\rangle > 0$.

To demonstrate, in Figure 2, the *starting states* are $s_0, s_3, s_4, s_5, s_8, g_0$, and g_1; the set of *connected states* of s_0 is $\mathbf{C}_0 = \{s_0, s_1, s_2\}$ and the set of *connected states* of s_4 is $\mathbf{C}_4 = \{s_4, s_6, s_7\}$; the *connected observation* of $\tilde{\mathbf{O}}(\{s_4, s_5\}) = \{s_4, s_5, s_6, s_7\}$; and the *ending states* are $s_2, s_3, s_6, s_7, s_9, g_0$, and g_1. Note that a state can be starting and ending state at the same time.

Definition 9 (Predecessor) *Given two connected observations $\tilde{\mathbf{O}}(\mathbf{S}')$ and $\tilde{\mathbf{O}}(\mathbf{S}'')$, $\tilde{\mathbf{O}}(\mathbf{S}')$ is the predecessor of $\tilde{\mathbf{O}}(\mathbf{S}'')$ if* $\exists a \in \mathbf{A}, s \in \tilde{\mathbf{O}}(\mathbf{S}'), s' \in \tilde{\mathbf{O}}(\mathbf{S}'') : \mathbf{T}\langle s, a, s'\rangle > 0 \wedge N(s) \neq N(s')$.

In Figure 2, $O_1 = \tilde{\mathbf{O}}(\{\mathbf{s_0}\})$ is a *predecessor* of O_2, O_3, O_4, and O_6.

To construct the augmented MDP for a POS-GRD problem, we start by augmenting states, actions, and transitions from the original MDP $\langle \mathbf{S}, s_0, \mathbf{A}, \mathbf{T}, \mathbf{C}, \mathbf{G}\rangle$, that is, we generate $\langle \tilde{\mathbf{S}}, \tilde{s}_0, \tilde{\mathbf{A}}, \tilde{\mathbf{T}}\rangle$ following the procedure described by Wayllace *et al.* [2017]. It is important to point out that when a state s is augmented to a state $\tilde{s} = \langle s, \mathbf{G}'\rangle$, $\tilde{s}$ projects the same observation as s, that is, $N(s) = N(\langle s, \mathbf{G}'\rangle)$ for any set $\mathbf{G}' \subseteq \mathbf{G}$.

4.1 Augmented MDPs for POS-GRD

For example, for the original MDP shown in Figure 3(a), the corresponding augmented states and actions are shown in Figure 3(b). Note that all the augmented states generated from the same state (e.g., s_2 or s_4) have the same observation. Every state is augmented with the set of possible goals for that state and every action is augmented with the set of goals for which that action is optimal (non-optimal actions in gray are not taken into account); the set of possible goals of a successor is found by intersecting the set of possible goals of its predecessor with the set of possible goals of the action executed to transition to that state.

If all states and actions were observable in the example depicted in Figure 3, the agent would reveal its goal as soon as one action is executed (since a_1 is an optimal action only for goal g_1 and a_0 is optimal only for goal g_0). However, in our partially-observable setting, where actions are not observable and all connected states have the same observation, the agent is able to hide its true goal for longer. For example, in Figure 3(b), when the observer observes $\langle O_1, O_2\rangle$, it could be due to the agent executing action a_1 from state s_0 and transitioning to state s_2 or it could be due to the agent executing action a_0 and transitioning to the same state s_2. Even though the two actions are optimal actions for two different goals g_1 and g_2, the observer is not able to distinguish between them. As a result, the agent is able to hide its true goal after executing either of those two actions if it transitions to state s_2. However, if the observer observes $\langle O_1, O_4\rangle$, then it knows with certainty that the agent executed action a_1 and can infer that the agent's goal is g_1.

To do this type of inference, we keep track of the set of possible goals with respect to the projected observations by extending Definitions 5 to 9 to the augmented domain. The extensions are trivial – in every definition, it is sufficient to substitute any reference to a state, action, or transition with their augmented counterpart. For instance, Definition 5 is extended as follows:

Definition 10 (Augmented Starting State) *Augmented state $\tilde{s}$ is an augmented starting state if* $\exists \tilde{s}' \in \tilde{\mathbf{S}}, \tilde{a} \in \tilde{\mathbf{A}} : \mathbf{T}\langle \tilde{s}', \tilde{a}, \tilde{s}\rangle > 0 \wedge N(\tilde{s}) \neq N(\tilde{s}') \vee \tilde{s} = \tilde{s}_0$, *where $\tilde{s}_0$ is the augmented initial state.*

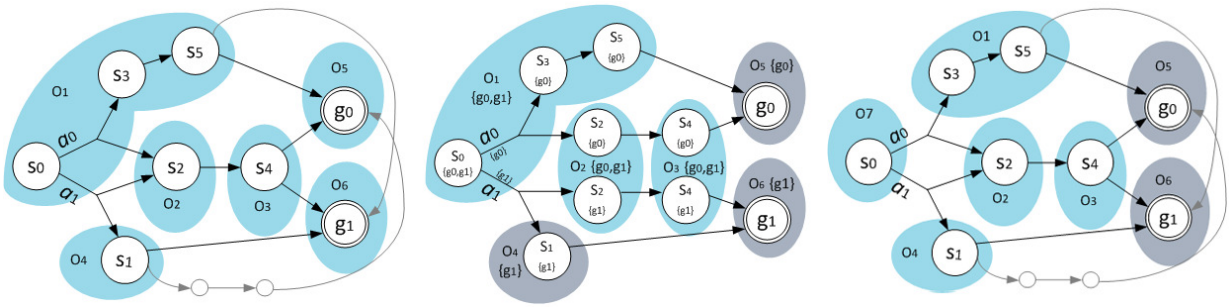
Definition 11 (Augmented Observation) *An augmented observation is a tuple $\langle \tilde{\mathbf{O}}(\tilde{\mathbf{S}}'), \mathbf{G}'\rangle$ where $\tilde{\mathbf{O}}(\tilde{\mathbf{S}}')$ is a connected observation and $\mathbf{G}' = \bigcup_{\langle s, \mathbf{G}''\rangle \in \tilde{\mathbf{S}}'} \mathbf{G}''$.*

Once the tuple $\langle \tilde{\mathbf{S}}, \tilde{s}_0, \tilde{\mathbf{A}}, \tilde{\mathbf{T}}\rangle$ is built, a structure of augmented observation sets (**AOS**) is constructed following Algorithm 1. This structure implements the model represented by the shaded regions in Figure 3(b). Specifically, each shaded region corresponds to one augmented observation and the predecessor for any augmented observation in **AOS** can be easily found.

Two connected observations can have states projecting the same observation. For example, given $\tilde{\mathbf{O}}(\tilde{\mathbf{S}}')$ and $\tilde{\mathbf{O}}(\tilde{\mathbf{S}}'')$, if $\tilde{\mathbf{S}}'' \cap \tilde{\mathbf{S}}' \neq \emptyset$, then by Definition 7, $\forall \tilde{s_i} \in \tilde{\mathbf{S}}', \tilde{s_j} \in \tilde{\mathbf{S}}'' : N(\tilde{s_i}) = N(\tilde{s_j})$. If their augmented observations are $\langle \tilde{\mathbf{O}}(\tilde{\mathbf{S}}'), \mathbf{G}'\rangle$ and $\langle \tilde{\mathbf{O}}(\tilde{\mathbf{S}}''), \mathbf{G}''\rangle$ respectively, and $\mathbf{G}' \neq \mathbf{G}''$, then both augmented observations are considered different. Since one state should belong to only one augmented observation, we need to create duplicates from all common augmented states and modify their transition functions accordingly. The function *Create* (line 13) will create a new augmented state $\tilde{s}'$ every time that an explored state $\tilde{s}$ belongs to another augmented observation.

Once Algorithm 1 is executed, the augmented observations in **AOS** contain the information that is available to the observer. Therefore, we can now use it to define a new augmented MDP that will allow us to correctly solve the initial goal recognition model (Definition 4).

Definition 12 (Augmented MDP for POS-GRD) *For a POS-GRD problem $PO = \langle D, \mathbf{G}, N\rangle$ with domain information $D = \langle \mathbf{S}, s_0, \mathbf{A}, \mathbf{T}, \mathbf{C}\rangle$ and the set of augmented observations* **AOS** *built following Algorithm 1, an augmented MDP is defined by a tuple $\langle \tilde{\mathbf{S}}, \tilde{s}_0, \tilde{\mathbf{A}}, \tilde{\mathbf{T}}, \tilde{\mathbf{C}}, \tilde{\mathbf{G}}\rangle$ that consists of the following:*



(a) Original Example (b) Augmented Observation Sets (c) After Sensor Refinement

Figure 3: Example

- *a set* $\tilde{\mathbf{S}}$ *of augmented states* $\tilde{s}$*, where* $\tilde{s} \in \tilde{\mathbf{S}} \iff \forall \langle \tilde{\mathbf{O}}(\tilde{\mathbf{S}}'), \mathbf{G}' \rangle \in \mathbf{AOS} : \tilde{s} \in \tilde{\mathbf{O}}(\tilde{\mathbf{S}}')$*;*
- *an augmented start state* $\tilde{s_0} = \langle s_0, \mathbf{G} \rangle$*;*
- *a set of augmented actions* $\tilde{\mathbf{A}} = \langle a, \mathbf{G}'' \rangle$*, where* $\mathbf{G}''$ *is the set of all goals for which* $a \in \mathbf{A}$ *is an optimal action;*
- *a transition function* $\tilde{\mathbf{T}} : \tilde{\mathbf{S}} \times \tilde{\mathbf{A}} \times \tilde{\mathbf{S}} \to [0, 1]$ *that gives the probability* $\tilde{T}(\langle s, \mathbf{G}' \rangle, \langle a, \mathbf{G}'' \rangle, \langle s', \mathbf{G}' \cap \mathbf{G}'' \rangle)$ *of transitioning from augmented state* $\langle s, \mathbf{G}' \rangle$ *to augmented state* $\langle s', \mathbf{G}' \cap \mathbf{G}'' \rangle$ *when augmented action* $\langle a, \mathbf{G}'' \rangle$ *is executed; this probability equals* $T(s, a, s')$ *if* $|\mathbf{G}' \cap \mathbf{G}''| > 1$ *and equals 0 otherwise;*
- *a cost function* $\tilde{\mathbf{C}} : \tilde{\mathbf{S}} \times \tilde{\mathbf{A}} \times \tilde{\mathbf{S}} \to \mathbb{R}^+$ *that gives cost* $\tilde{C}(\tilde{s}, \tilde{a}, \tilde{s}') = 0$ *if* $\exists \langle \tilde{\mathbf{O}}(\tilde{\mathbf{S}}'), \mathbf{G}^o \rangle \in \mathbf{AOS} : \tilde{s}' \in \tilde{\mathbf{O}}(\tilde{\mathbf{S}}') \wedge |\mathbf{G}^o| \leq 1$ *and* $\tilde{C}(\tilde{s}, \tilde{a}, \tilde{s}') = C(s, a, s')$ *otherwise; and*
- *the set of augmented goals* $\tilde{\mathbf{G}} \subset \tilde{\mathbf{S}} = \{\tilde{s} \mid \forall \langle \tilde{\mathbf{O}}(\tilde{\mathbf{S}}'), \mathbf{G}^o \rangle \in \mathbf{AOS} : |\mathbf{G}^o| = 1 \wedge \tilde{s} \in \mathbf{S}'\}$

4.2 Computing the *wcd*

Function Create($\tilde{\mathbf{S}}_s, \mathbf{G}', \mathbf{AOS}$)

$\tilde{\mathbf{S}}'_s \leftarrow \emptyset$
foreach $\tilde{s} \in \tilde{\mathbf{S}}_s$ **do**
  if $\exists \tilde{\mathbf{O}}(\tilde{\mathbf{S}}) \in \mathbf{AOS} : \tilde{s} \in \tilde{\mathbf{O}}(\tilde{\mathbf{S}})$ **then**
    $\tilde{s}' \leftarrow newState(\tilde{s})$
    $\tilde{\mathbf{S}}'_s \leftarrow \tilde{\mathbf{S}}'_s \cup \tilde{s}'$
    $\tilde{\mathbf{S}} \leftarrow \tilde{\mathbf{S}} \cup \tilde{s}'$
  else
    $\tilde{\mathbf{S}}'_s \leftarrow \tilde{\mathbf{S}}'_s \cup \tilde{s}$
$\tilde{\mathbf{O}}(\tilde{\mathbf{S}}') \leftarrow connectedObs(\tilde{\mathbf{S}}')$
return $\langle \tilde{\mathbf{O}}(\tilde{\mathbf{S}}'), \mathbf{G}' \rangle \leftarrow augmentedObs(\tilde{\mathbf{O}}(\tilde{\mathbf{S}}'), \mathbf{G}')$

Definition 13 (*wcd*) *The wcd of a POS-GRD problem PO is defined as:*

$$wcd(PO) = \max_{\tilde{\pi} \in \tilde{\mathbf{\Pi}}} V_{\tilde{\pi}}(\tilde{s}_0) \tag{2}$$

$$V_{\tilde{\pi}}(\tilde{s}) = \sum_{\tilde{s}' \in \tilde{\mathbf{S}}} \tilde{T}(\tilde{s}, \tilde{\pi}(\tilde{s}), \tilde{s}') \big[\tilde{C}(\tilde{s}, \tilde{\pi}(\tilde{s}), \tilde{s}') + V_{\tilde{\pi}}(\tilde{s}') \big] \tag{3}$$

where $\tilde{\mathbf{\Pi}}$ *is the set of augmented policies in the augmented MDP as specified in Definition 12 and* $V_{\tilde{\pi}}(\tilde{s}_0)$ *is the expected cost for* s_0 *with augmented policy* $\tilde{\pi}$ *computed recursively using Equation 3.*

The baseline algorithm to compute the *wcd* is to follow Equations 2 and 3, that is, for each possible augmented policy $\tilde{\pi}$, run a VI-like algorithm, where instead of using the Bellman equation (Equation 1) we use Equation 3, run the algorithm until convergence, and store the value $V_{\tilde{\pi}}(\tilde{s}_0)$ to find the maximum among all policies. Finding the maximum expected cost should be done in one execution of the algorithm using the following equation:

$$V^*(\tilde{s}) = \max_{\tilde{a} \in \tilde{A}} \sum_{\tilde{s}' \in \tilde{\mathbf{S}}} \tilde{T}(\tilde{s}, \tilde{a}, \tilde{s}') \big[\tilde{C}(\tilde{s}, \tilde{a}, \tilde{s}') + V^*(\tilde{s}') \big] \tag{4}$$

Observe that this equation is equivalent to Equations 2 and 3, and it differs from the Bellman equation only in the operator: this one uses the maximization instead of minimization. The main problem of maximizing costs is the existence of infinite loops since the optimal policy is to accumulate cost infinitely. The augmented MDP does not have infinite loops because the augmented actions are constructed using only optimal actions to arrive to any possible goal, hence, the only case of a cycle could be if an agent transitioning from state $\tilde{s}$ to $\tilde{s}'$ executes an augmented action $\langle \tilde{a}, \mathbf{G}' \rangle$ and to transition from $\tilde{s}'$ to $\tilde{s}$ executes action $\langle \tilde{a}', \mathbf{G}'' \rangle$, where $\mathbf{G}' \cap \mathbf{G}'' = \emptyset$, that is, if both actions are optimal for a different set of possible goals. However, this is impossible because the set of possible goals of an augmented state is always a subset of the set of possible goals of its predecessors. A formal sketch proof of this property was presented by Wayllace *et al.* [2017] for the augmented MDP for S-GRD problems. The property remains true for POS-GRD since the new augmented states are duplicates of others; therefore, the possible augmented actions, successors and predecessors remain the same.

Therefore, a VI-like algorithm using Equation 4 can be used. Even further, we took advantage of the structure of the augmented MDP that usually allows to group

Algorithm 1: CONSTRUCTION OF THE AUGMENTED STATE SPACE

Input: $\langle \tilde{\mathbf{S}}, \tilde{s}_0, \tilde{\mathbf{A}}, \tilde{\mathbf{T}} \rangle$
$\mathbf{AOS} \leftarrow \{\langle \tilde{\mathbf{O}}(\{\tilde{\mathbf{s}}_0\}), \mathbf{G} \rangle\}$
$Q \leftarrow \tilde{\mathbf{O}}(\{\tilde{\mathbf{s}}_0\})$
while $Q \neq \emptyset$ **do**
  $\tilde{\mathbf{O}}(\tilde{\mathbf{S}}_i) \leftarrow$ Q.dequeue()
  Find all ending states $s_e \in \tilde{\mathbf{O}}(\tilde{\mathbf{S}}_i)$
  foreach $s_e \in \tilde{\mathbf{O}}(\tilde{\mathbf{S}}_i)$ **do**
    if $\exists \tilde{a} \in \tilde{\mathbf{A}}$ **then**
      Find all starting states $\tilde{s}_s$ with $\mathbf{T}(\tilde{s}_e, \tilde{a}, \tilde{s}_s) > 0 \wedge N(\tilde{s}_e) \neq N(\tilde{s}_s)$
      Group all $\tilde{s}_s$ with same $N(\tilde{s}_s)$
    foreach *group* $\mathbf{S}'$ **do**
      if $\nexists \langle \tilde{\mathbf{O}}(\tilde{\mathbf{S}}'), \mathbf{G}' \rangle \in \mathbf{AOS} : \mathbf{G}' = \bigcup_{\langle s_s, \mathbf{G}'' \rangle \in \tilde{\mathbf{S}}'} \mathbf{G}''$ **then**
        $\langle \tilde{\mathbf{O}}(\tilde{\mathbf{S}}'), \mathbf{G}' \rangle \leftarrow$ Create $(\{\tilde{s}_s | \tilde{s}_s \in \tilde{\mathbf{S}}'\}, \mathbf{G}', \mathbf{AOS})$
        if $\nexists \langle \tilde{\mathbf{O}}(\tilde{\mathbf{S}}''), \mathbf{G}' \rangle \in \mathbf{AOS} : \tilde{\mathbf{S}}' \neq \tilde{\mathbf{S}}'' \wedge \tilde{\mathbf{O}}(\tilde{\mathbf{S}}') = \tilde{\mathbf{O}}(\tilde{\mathbf{S}}'')$ **then**
          $\mathbf{AOS} \leftarrow \mathbf{AOS} \cup \langle \tilde{\mathbf{O}}(\tilde{\mathbf{S}}'), \mathbf{G}' \rangle$
          Enqueue $\langle \tilde{\mathbf{O}}(\tilde{\mathbf{S}}'), \mathbf{G}' \rangle$ in Q

augmented states into *strongly connected components* (SCC), therefore, algorithms similar to TVI can be used. The Tarjan's algorithm [Tarjan, 1972] was used to form SCCs. Once the SCCs are constructed, a VI-like algorithm is executed on each SCC in reverse topological order.

It is worth mentioning that non-reachable states in the augmented MDP can be prunned, as well as states that belong to an augmented observation whose predecessors have only one possible goal.

4.3 Reducing the *wcd*

Once the *wcd* of the model has been computed, we propose to apply two types of modifications with the objective to reduce the *wcd*: (1) Sensor refinement and (2) Action removal.

Sensor Refinement: In a POS-GRD problem, partial observability is due to low-resolution sensors that make it impossible for an observer to distinguish a number of similar states. As a result, all the states covered by a sensor have the same observation and the observer does not know which is the actual state of the agent. By refining sensor resolution, the observer can gain better observability of the states covered by a particular sensor. The design objective is thus to identify which sensors refine such that the resulting *wcd* is minimized under the specified constraints. Ideally, all sensors should be refined so that the all states are fully observable as this will guarantee that the *wcd* is minimized. However, we assume that there is a limited budget available and only a limited number of sensors can be refined.

In this paper, we use a simple implementation of sensor refinement (as defined in Definition 3 by allowing making a single state fully observable. In other words, if the agent is in any of the *refined states*, the observer is able to observe it with full certainty. Figure 3(c) shows an example of sensor refinement for the original example in Figure 3(a), where state s_0, previously mapped to the same observation O_1 as states s_3 and s_5, is now mapped to a new observation O_7. This makes it possible to distinguish s_0 from the other two states. In our setting, there is a maximum of k sensor refinement modifications that can be performed.

Algorithm 2 describes the sensor refinement pseudocode for choosing the k states to *refine*. It first constructs a queue that contains all subsets of up to k states (line 27). Then, it iteratively evaluates each set by computing the *wcd* if those refined states (line 32). If the resulting *wcd* is smaller than the minimal *wcd* found so far, it updates the best *wcd* with that value and stores that set of states as the best set (lines 33-35). After evaluating all sets of states in the queue, it updates the sensor function N by replacing the observation of each of the k states with new observations (lines 36-39).

Action Removal: Another (more classical) modification that can be performed to the model is action removal, where there is also a constraint on the number of modifications. Specifically, the objective is to minimize the *wcd* by removing at most k actions. Similar to the algorithms previously used by Wayllace *et al.* [2017], we enumerate through all possible combinations of up to k actions, compute the *wcd* for each combination, and choose the combination that reduces the *wcd* the most.

Algorithm 2: Sensor Refinement

create a queue $Q \leftarrow$ combination of up to k states
$wcd^* \leftarrow \infty$
$\mathbf{S}^* \leftarrow \emptyset$
while $Q \neq \emptyset$ **do**
    $\mathbf{S}_{cand} \leftarrow Q$.deque
    $wcd_{cand} \leftarrow compute\_wcd(\mathbf{S}_{cand})$
    if $\text{wcd}_{cand} < \text{wcd}^*$ **then**
        $wcd^* \leftarrow wcd_{cand}$
        $\mathbf{S}^* \leftarrow \mathbf{S}_{cand}$
foreach $s_i \in \mathbf{S}^*$ **do**
    $\mathbf{O}_i^N = getObs(s_i)$
    $\mathbf{O}_i^N \leftarrow \mathbf{O}_i^N \setminus \{s_i\}$
    add mapping $s_i \rightarrow \mathbf{O}_{n+i}^N$ to sensor function N

5 Empirical Evaluation

The domain used to run the experiments is a modification of the domain called ROOM, which was used in the Non-Deterministic Track of the 2006 ICAPS International Planning Competition.[1] It is a grid world where the actions as well as the transition probabilities are defined individually for each state. Each instance of this domain is defined by the x- and y-dimensions of the room and the number of possible goals. The initial setting for partial observability was added, specifically, four contiguous states were mapped to the same observation.

Three type of experiments were performed: (1) Partial observability with sensor refinement (SR); (2) Partial observability with action removal (AR); and (3) Full observability of states with action removal (FO), (note that this is different to S-GRD since the actions are non-observable here). The smaller instances used a budget k=2 in all settings while the larger ones used k=1; they timed out with k=2. The parameter k represents the maximum number of states to refine in SR and the maximum number of actions to remove in AR and FO. Both SR and AR start with the same initial partially-observable problem. The only difference is in the type of modifications allowed. FO assumes that all states are fully observable and only the actions are hidden to the observer. The experiments were conducted on a 3.1 GHz Intel Core i7 with 16 GB of RAM and a timeout of two days was imposed.

We make the following observations:

- The initial *wcd* is larger for all the instances with partially-observable states (SR and AR) compared to if they are fully observable (FO). However, the ratio between both values differs across instances, which is interesting because it shows that not only the resolution (number of states covered by one sensor), but also the placement (which states are covered by the same sensor) of sensors matters. Thus, in the future, we plan to conduct additional experiments to optimize sensor placement, even without improving their resolution.
- The *wcd* reduced for all instances when we applied sensor refinement. However, the *wcd* reduced for only one instance when we applied action removal (two others also show some reduction, but it might be due to rounding errors). This is because the domain in general has few policies that are common to more than one goal and removing actions increases the initial expected cost or causes the goal to become unreachable.
- In SR, only two instances were able to match the *wcd* value of FO after reduction, but four other instances were close. This does not depend on the size of the state space (one match occurred for instance 4-4-3 and the other for 32-32-3), which also suggests that the initial sensor mapping could affect the quality of goal recognition.
- The running time grows exponentially with the size of the reachable state space and, as expected, with the number of modifications k.

[1] http://idm-lab.org/wiki/icaps/ipc2006/probabilistic/

6 Conclusions and Future Work

Previous work in GRD did not account for partially-observable states, which is relevant to many applications such as agent navigation, where only the current state is observable, not the intention of movement. Additionally, observations depend on the resolution of the sensor. Thus, some states can be perceived as identical to other states. In response to these observations, this paper proposes the *Partially Observable S-GRD* (POS-GRD) problem where (1) actions are not observable and (2) states are partially observable. New algorithms taking partial observability into account to compute the *wcd* and to perform sensor refinement in POS-GRD problems were proposed. Experimental results show that sensor refinement always reduces the *wcd* and suggest that the initial sensor configuration affects the reduction ratio when the number of possible modifications is limited.

Future work includes the use of heuristics to prune the search space for higher values of k. Since the modifications start from 1 to k, the idea is to find all augmented states that have less or equal expected cost than the current (minimized) value of *wcd* and prune the rest of the searching space. We are also interested in use other metrics and design mechanisms in the POS-GRD context.

Acknowledgments

This research is partially supported by NSF grant 1540168. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

References

[Bellman, 1957] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

Domain Instances	k	Sensor Refinement (SR)		Action Removal (AR)		Full State Observability (FO)	
		wcd Reduction	Runtime (s)	*wcd* Reduction	Runtime (s)	*wcd* Reduction	Runtime (s)
4-4-3	2	4.98 → 3.71	0.25	4.98 → 4.98	0.18	3.71 → 3.71	0.26
8-8-2	2	16.05 → 16.03	19.76	16.05 → 16.05	23.48	16.02 → 16.02	22.65
8-8-3	2	10.64 → 9.22	51.78	10.64 → 10.64	27.96	9.09 → 9.09	13.18
12-12-3	2	16.67 → 16.53	231.48	16.67 → 16.67	102.05	16.42 → 16.42	82.26
16-16-3	2	16.71 → 8.16	3,238.08	16.71 → 16.71	1,538.90	6.62 → 6.62	955.53
20-20-3	2	53.42 → 40.27	14,595.43	53.42 → 53.33	28,649.10	38.59 → 38.59	5,845.37
24-24-3	2	19.28 → 12.61	53,846.64	19.28 → 19.28	8,322.37	12.15 → 12.15	1,948.80
32-32-2	1	79.67 → 48.17	395.39	79.67 → 79.50	481.14	39.28 → 38.91	105.14
32-32-3	1	87.57 → 86.57	455.83	87.57 → 87.57	632.53	86.57 → 86.57	145.22
44-44-3	1	92.74 → 87.21	1,519.19	39.28 → 39.28	13.18	73.76 → 73.76	1,116.03

Table 2: Experimental Results

[Carberry, 2001] Sandra Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11:31–48, 2001.

[Dai *et al.*, 2011] Peng Dai, Mausam, Daniel S Weld, and Judy Goldsmith. Topological value iteration algorithms. *Journal of Artificial Intelligence Research*, 42:181–209, 2011.

[Keren *et al.*, 2014] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 154–162, 2014.

[Keren *et al.*, 2015] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design for non-optimal agents. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3298–3304, 2015.

[Keren *et al.*, 2016a] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design with non-observable actions. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3152–3158, 2016.

[Keren *et al.*, 2016b] Sarah Keren, Avigdor Gal, and Erez Karpas. Privacy preserving plans in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3170–3176, 2016.

[Keren *et al.*, 2018] Sarah Keren, Avigdor Gal, and Erez Karpas. Strong stubborn sets for efficient goal recognition design. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2018.

[Mausam and Kolobov, 2012] Mausam and Andrey Kolobov. *Planning with Markov Decision Processes: An AI Perspective*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[Ramírez and Geffner, 2010] Miquel Ramírez and Hector Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[Son *et al.*, 2016] Tran Cao Son, Orkunt Sabuncu, Christian Schulz-Hanke, Torsten Schaub, and William Yeoh. Solving goal recognition design using ASP. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3181–3187, 2016.

[Sukthankar *et al.*, 2014] Gita Sukthankar, Christopher Geib, Hung Hai Bui, David Pynadath, and Robert P Goldman. *Plan, activity, and intent recognition: Theory and practice*. Newnes, 2014.

[Tarjan, 1972] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[Wayllace *et al.*, 2016] Christabel Wayllace, Ping Hou, William Yeoh, and Tran Cao Son. Goal recognition design with stochastic agent action outcomes. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3279–3285, 2016.

[Wayllace *et al.*, 2017] Christabel Wayllace, Ping Hou, and William Yeoh. New metrics and algorithms for stochastic goal recognition design problems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4455–4462, 2017.