Observation Decoding with Sensor Models: Recognition Tasks via Classical Planning

Diego Aineto and **Sergio Jimenez** and **Eva Onaindia**
Valencian Research Institute for Artificial Intelligence
Universitat Politècnica de València
{dieaigar, serjice, onaindia}@vrain.com

Abstract

Observation decoding aims at discovering the underlying state trajectory of an acting agent from a sequence of observations. This task is at the core of various recognition activities that exploit planning as resolution method but there is a general lack of formal approaches that reason about the partial information received by the observer or leverage the distribution of the observations emitted by the sensors. In this paper, we formalize the observation decoding task exploiting a probabilistic sensor model to build more accurate hypothesis about the behaviour of the acting agent. Our proposal extends the expressiveness of former recognition approaches by accepting observation sequences where one observation of the sequence can represent the reading of more than one variable, thus enabling observations over actions and partially observable states simultaneously. We formulate the probability distribution of the observations perceived when the agent performs an action or visits a state as a classical cost planning task that is solved with an optimal planner. The experiments will show that exploiting a sensor model increases the accuracy of predicting the agent behaviour in four different contexts.

Introduction

In recent years, the interest in recognition tasks has grown in the planning community. Plan recognition, and in particular the approach presented in (Ramírez and Geffner 2010), is probably the most notable example among the recognition tasks related to planning. This work showed that predicting the goals and plans of an acting agent can be addressed via planning, transforming the recognition task to plan generation. This reduction-to-planning scheme, which uses a planner to fill the gaps in an incomplete observation sequence, was later adopted by many other recognition tasks; e.g., goal recognition design (GRD) (Keren, Gal, and Karpas 2014), counter-planning (Pozanco et al. 2018) or model recognition (Aineto et al. 2019).

A key component of a recognition task is the definition of the partial information received by the observer. Formally, a *sensor model* is a function that specifies how likely it is to perceive some observation from a given state. Vision sensors, for instance, are more likely to provide unreliable readings in the absence of light, while the readings of temperature sensors become more erroneous at extreme temperatures. The common assumption of the approaches cited above is, however, to ignore the distribution of the observations, thus assuming all observations are uniformly distributed. In general, little attention has been paid to formalizing the sensing model of the observer, and most works adopt a simplistic view that disregards how the input observations are produced. This is also the case of approaches to goal recognition in non-deterministic environments that reduce the sensor model of the observer to a Boolean problem according to whether the execution of the agent complies with the observation or not (Ramírez and Geffner 2011). The controlled observability planning problem defined in (Kulkarni, Srivastava, and Kambhampati 2019) introduces a deterministic observer sensor model able to produce the same observation for different state-action pairs. More recently, some researchers define noisy (non-deterministic) sensor models that emit one among several possible tokens for an executed action in the context of GRD (Keren, Gal, and Karpas 2019). It is worth noting that all these cited latter works only handle observations over actions.

In this work we build upon the ideas presented in (Ramírez and Geffner 2010), to posit the problem of *observation decoding*, which consists in finding the most probable state trajectory for a given observation sequence. Our extension exploits rich *sensor models* to build more informed hypothesis about the behaviour of the acting agent. In doing so, we also address one of the more limiting factors of (Ramírez and Geffner 2010) which has been inherited by the derived approaches, the single fluent observations. This limitation has meant that so far, the observer agent was only able to observe one single element at a time, be it an action or a state variable (Sohrabi, Riabov, and Udrea 2016). This new understanding of the core task enables the formulation of a general framework capable of recognizing any combination of the elements involved in the execution of an agent, namely the states, actions or goals.

Figure 1 shows an scenario where exploiting a sensor model helps us make more accurate predictions about the unobserved actions of an agent that is navigating a 5×5 grid. According to its action model, we know the agent can only move one cell at a time, in any of the four directions N,S,W or E, and all moves have a unitary cost. In this example we

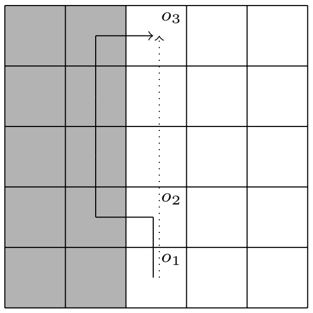

Figure 1: A three-observation sequence O=(o_1, o_2, o_3) of the actual location of an agent in a 5×5 grid, and two plans (solid and dotted arrows) that might generate the given observation sequence.

have *open* tiles (in white) and *covered* tiles (grey), and we use a zenith camera to observe the agent. We also have a *sensor model* that specifies that the camera can pinpoint the location of the agent with 90% reliability when the agent is on an *open* tile, but it is unable to locate the agent when it is moving through the *covered* tiles (e.g. because there is no line of sight). The observation sequence collected by the camera is denoted by (o_1, o_2, o_3). The solid and dotted arrows represent two hypothesis on the agent trajectory that may be induced from the observations. Taking into account just the observations and the action model, the most rational hypothesis is that the agent followed the dotted arrow from o_1 to o_3. However, if we consider a richer planning model that includes also what we know about the camera (our sensor in this example), we would claim that it is more likely the agent follows the trajectory represented by the solid arrow.

Probabilistic sensor models are commonly used for plan generation of an acting agent in partially observable environments like in POMDPs (Kaelbling, Littman, and Cassandra 1998; Silver and Veness 2010). They are also an indispensable component of generative models like HMMs and DBNs (Ghahramani 2001) for uncovering states in predictive tasks. Actually, exploiting a probabilistic sensor model in PR can be assimilated to an inference task under a probabilistic model. Bayesian inference techniques have shown to work well for PR in human activities (Nazerfard and Cook 2015) as well as other techniques such as topic modeling (Freedman, Jung, and Zilberstein 2014). However, they are not particularly well suited for partial input observations or decoding gapped sequences observed at randomly chosen instants. While Hidden semi-Markov models are proposed to address missing observations associated to state transition in probabilistic models (Yu 2016), our proposal relies on exploiting planning models alongside a probabilistic *sensor model* to determine how likely an observation is. Thus, our proposal can be regarded as an approach that combines the predictive power of linear-time generative models with the expressive power of a planning formalism.

In summary, the specific contributions of this paper are:

1. The formalization of the *observation decoding* task that leverages a probabilistic *sensor model* to build more accurate hypothesis about the behaviour of the acting agent.
2. A method for reducing *observation decoding* to plan generation and solving it with an optimal planner.
3. Our model accepts observation sequences where each observation can be a set of observable tokens. The classical interpretation of this is that the observer agent may perceive both actions and partial states. The model also supports a different set of variables (alphabet) for the observations.

Background

A *planning problem* is a tuple $P = \langle F, A, I, G\rangle$, where F is a finite set of fluent symbols that denote the state variables, A is a set of deterministic actions with preconditions $\mathsf{pre}(a)$ and effects $\mathsf{eff}(a)$, I is the initial state of the problem and G is the goal condition. $\mathcal{L}(F)$ is the set of all literal sets on F, i.e. all partial assignments of values to fluents, such that $\mathsf{pre}(a) \in \mathcal{L}(F)$ and $\mathsf{eff}(a) \in \mathcal{L}(F)$. A problem state s is a full assignment of values to fluents in which every fluent of F is assigned a value true (f) or false ($\neg f$); i.e. $|s| = |F|$. An action a is applicable in a state s if $\mathsf{pre}(a) \subseteq s$. The successor state is defined as $\theta(s, a) = \{s \setminus \neg\mathsf{eff}(a) \cup \mathsf{eff}(a)\}$, where $\neg\mathsf{eff}(a)$ is the complement of $\mathsf{eff}(a)$ to ensure that the successor state is a well-defined state with all state variables F set to true or false. We assume that the cost of executing an action $a \in A$ is given by the non-negative cost function $c(a)$. Given a planning problem $P = \langle F, A, I, G\rangle$, we define the tuple $\mathcal{M} = \langle F, A\rangle$ as the *planning model* of P.

A *plan* π for P is an action sequence $\pi = \langle a_1, \ldots, a_n\rangle$, and $|\pi| = n$ denotes the plan length. The execution of π in the initial state I of P induces a *trajectory* $\tau = \langle s_0, a_1, s_1, \ldots, a_n, s_n\rangle$ such that $s_0 = I$ and, for each $1 \leq i \leq n$, it holds that a_i is applicable in s_{i-1}, and $s_i = \theta(s_{i-1}, a_i)$. A plan π solves P if G holds in the last state of the induced trajectory τ; i.e., $G \subseteq s_n$. The cost of a plan $\pi = \langle a_1, \ldots, a_n\rangle$ is $c(\pi) = \sum_{i=1}^{n} c(a_i)$ and we say it is optimal if its cost is minimal.

Many tasks require input observations that represent the information which is actually observable when an agent executes a plan π for solving a problem P in a particular context. An observation sequence is denoted as $O = \langle o_1, \ldots, o_n\rangle$ where o_i represents the observation at the i-th step of π's execution.

In action model learning, it is commonly assumed that O is, at least, a fully-observable action sequence; that is, O contains all the actions of π done by the agent, possibly encoded as a fluent in each o_i. This is the case of well-known systems like ARMS (Yang, Wu, and Jiang 2007), SLAF (Amir and Chang 2008), LOUGA (Kucera and Barták 2018) or LOCM (Cresswell, McCluskey, and West 2013). When O is a complete action sequence (it may additionally contain observable fluents of the state trajectory), it means there are no gaps in O and so $|\pi|$ is known. Without entering into the discussion whether it is sensible to assume that the sensors reliably emit a signal for each executed action, a clear advantage of knowing the length of the plan is that it enables using bounded reasoning techniques. A notable exception to this assumption is FAMA (Aineto, Jiménez, and Onaindia 2019), which supports observations as minimal as $O = \langle o_1, o_2 \rangle$, where $o_1 = I$ and o_2 is an observation of the goal state.

The pioneering work on plan recognition presented in (Ramírez and Geffner 2009; Ramírez and Geffner 2010) accepts incomplete observation sequences ($|O| \leq |\pi|$), where each $o_i \in O$ is an action of A. This approach applies a transformation to compile away the observations by encoding them as extra fluents of F, extra actions of A and extra goals of G. A subsequent extension to this work (Sohrabi, Riabov, and Udrea 2016) addresses observations over fluents rather than actions, and handles unreliable observations in the form of noisy and missing observations. A limitation of this model is, however, that each o_i of O is constrained to be a single fluent of F.

The formulation of GRD in partially observable environments (Keren, Gal, and Karpas 2019) considers sensor models that account for non-observable actions, low-sensor resolution models that emit the same observation token for different actions, and noisy (non-deterministic) sensor models that emit one among several possible tokens for an executed action. In all these models, observable tokens are associated to observations over actions.

Our proposal also exploits sensor models that emit incomplete observation sequences, but unlike the above approaches each o_i of O is a set of observable tokens representing an action observation, observations over fluents or a combination of both. This way, we can handle different alphabets of symbols for the joint emission of observations over A and F.

The sensor model

We extend the classical planning model $\mathcal{M}_t = \langle F, A \rangle$ with a sensor model $\mathcal{M}_e$ that specifies the observations emitted (and perceived by the observer) from a given state.

We compactly represent a probabilistic sensor model with the pair $\mathcal{M}_e = \langle Y, \Phi \rangle$, where:

- $Y = \{Y_1, \ldots, Y_n\}$ is a set of *observable variables*; a variable $Y_i \in Y$ has a finite domain D_{Y_i} which values are given by the sensor reading. The symbol ϵ is used to represent the *empty sensor reading*.
- Φ is a set of functions $\Phi_{Y_i} : D_{Y_i} \times \mathcal{S} \to [0, 1], \forall Y_i \in Y$, that denote the probability of observing a token $v \in D_{Y_i}$ in a state $s \in \mathcal{S}$. The function Φ_{Y_i} is defined as a discrete probability distribution $P(Y_i = v | S = s)$ that maps a state s to a weighted disjunction of observable atoms $Y_i = v$. Since Φ_{Y_i} defines a probability distribution, it must hold that $\sum_{v \in D_{Y_i}} \Phi_{Y_i}(v, s) = 1$, for all $s \in \mathcal{S}$.

In practice, many states share the same probability distribution for the emission of the observable variables so it is often better to define Φ_{Y_i} for sets of states. A compact way to do this is to define a condition $c \in \mathcal{L}(F)$ so that $\mathcal{S}_c \subseteq \mathcal{S}$ is the subset of states such that $\forall s \in \mathcal{S}_c, c \subseteq s$. Thus, c determines the set of states $\mathcal{S}_c$ with the same observability conditions (e.g. the location of a robot is observable from a well-lit room). We will use this notation in the rest of the paper when the set of states $\mathcal{S}_c$ for a given condition c are uniquely identified.

This sensor model enables the emission of observations when the state representation, F, and the observable variables, Y, do not share the same alphabet; i.e., $Y \nsubseteq F$. This way, variables Y can be used to represent both observations over states and actions. Sensor models of this kind are used, for instance, for the derivation of *Finite State Controllers* (Bonet, Palacios, and Geffner 2009; 2010).

We now define the notion of observation supported in this work.

Definition 1 (Observation). *Given a planning model $\mathcal{M}_t = \langle F, A \rangle$ extended with a sensor model $\mathcal{M}_e = \langle Y, \Phi \rangle$, an observation $o = \langle Y_1 = v_1, \ldots, Y_n = v_n \rangle$ is a total assignment of the observable variables Y with values within $D_{Y_i} \cup \{\epsilon\}$ for each $Y_i \in Y$.*

Given $\mathcal{M}_t = \langle F, A \rangle$ extended with a sensor model $\mathcal{M}_e = \langle Y, \Phi \rangle$, an observation sequence is denoted as $O = \langle o_1, o_2, \ldots, o_m \rangle$, where each $o_i \in O$ is an observation. Since O can be an incomplete observation sequence, the total number of states traversed by the acting agent may be unknown. In the example of Figure 1 only three observations are available $\langle o_1, o_2, o_3 \rangle$ but, according to the given action model, more than three states are actually visited for moving from a state that emits o_1 to a state that emits o_3.

Following, we present a detailed explanation of the sensor model of Figure 1, and an outline of the sensor model for a *blocksworld* domain with two hands.

Grid domain. An observation o in the grid example of Figure 1 is the assignment of one observable variable $\mathsf{loc_{x,y}}$ associated to the fluents $\{\mathsf{loc_{i,j}} | 1 \leq i, j \leq 5\}$, which represents the actual agent location in a cell $(i.j)$.

1. When $\mathsf{loc_{1,j}}$ or $\mathsf{loc_{2,j}}$ (the agent is at any grey tile), the observation of $\mathsf{loc_{x,y}}$ is formalized by:
   $P(\mathsf{loc_{x,y}} = (\mathsf{i,j}) | \mathsf{loc_{i,j}}) = 0$
   $P(\mathsf{loc_{x,y}} = \epsilon | \mathsf{loc_{i,j}}) = 1$
2. When $\mathsf{loc_{3,j}}$, $\mathsf{loc_{4,j}}$, or $\mathsf{loc_{5,j}}$ (the agent is at any white tile), the observation of $\mathsf{loc_{x,y}}$ is formalized by:
   $P(\mathsf{loc_{x,y}} = (\mathsf{i,j}) | \mathsf{loc_{i,j}}) = 0.9$
   $P(\mathsf{loc_{x,y}} = \epsilon | \mathsf{loc_{i,j}}) = 0.1$

Note that our sensor model allows for observations with *missing data* (when o is $\mathsf{loc_{x,y}} = \epsilon$), and observations with *noisy data* if the value of the observable variable $\mathsf{loc_{x,y}}$ is different from the actual agent location given by $\mathsf{loc_{i,j}}$.

Blocksworld domain with two hands. An observation o in this domain consists of the assignment of two observable variables, $\mathsf{hold_x}$ and $\mathsf{hold_y}$, associated to the fluents $\{\mathsf{hold_{h1,b_i}} | \mathsf{b_i} \in \{\mathsf{b1, b2}\}\}$ and $\{\mathsf{hold_{h2,b_i}} | \mathsf{b_i} \in \{\mathsf{b1, b2}\}\}$, respectively, which represent whether each hand $\mathsf{h_i}$ is grabbing a block $\mathsf{b_i}$ in a state (in this example there is only two blocks, b_1 and b_2). Let's assume the emission of the two observable variables $\mathsf{hold_x}$ and $\mathsf{hold_y}$, such that $\mathsf{h1}$ has higher observability than $\mathsf{h2}$:

$P(\mathsf{hold_x} = \mathsf{b_i} | \mathsf{hold_{h1,b_i}}) = 0.7 \quad P(\mathsf{hold_x} = \epsilon | \mathsf{hold_{h1,b_i}}) = 0.3$
$P(\mathsf{hold_y} = \mathsf{b_i} | \mathsf{hold_{h2,b_i}}) = 0.6 \quad P(\mathsf{hold_y} = \epsilon | \mathsf{hold_{h2,b_i}}) = 0.4$

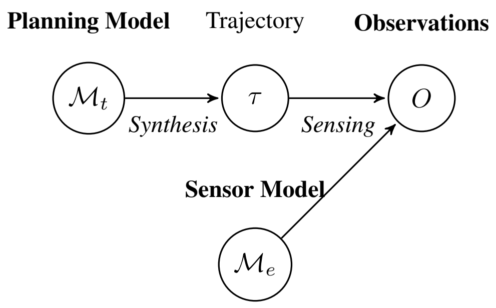

Figure 2: *Bayesian network* illustrating the dependencies between the planning model, the sensor model, and the input sequence of observations.

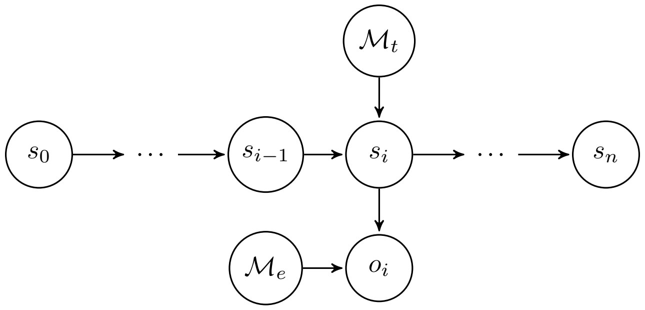

Figure 3: *Dynamic Bayesian network* that, for time-stamp i, unfolds the relations between a trajectory, the action and sensor models, and the observation of the trajectory.

Observation decoding with sensor models

This section formalizes the problem of finding the most probable state trajectory of an agent whose action model is given by $\mathcal{M}_t = \langle F, A\rangle$ from an observation sequence O generated with $\mathcal{M}_e = \langle Y, \Phi\rangle$.

Figure 2 shows a *Bayesian network* that illustrates the relations between a given planning model $\mathcal{M}_t$, a sensor model $\mathcal{M}_e$, and a given input sequence of observations $O = \langle o_1, o_2 \ldots, o_m\rangle$. *Synthesis* refers to the generation of a trajectory $\tau = \langle s_0, a_1, s_1, \ldots, a_n, s_n\rangle$ with the model $\mathcal{M}_t$. *Sensing* refers to the observation of a trajectory τ with the sensors modeled by $\mathcal{M}_e$. Taking into account that both trajectories and sequences of observation are time-stamped sequences, the *Bayesian network* of Figure 2 can be unfolded into a *Dynamic Bayesian Network* (Figure 3) to formalize the *synthesis probability* and the *sensing probability* functions.

The synthesis probability. Given a planning model $\mathcal{M}_t = \langle F, A\rangle$ and a trajectory $\tau = \langle s_0, s_1, \ldots, s_n\rangle$, the *synthesis probability* is the probability of generating τ with $\mathcal{M}_t$:

$$P(\tau|\mathcal{M}_t) = P(s_0)\prod_{i=1}^{|\tau|} P(s_i|s_{i-1}, \mathcal{M}_t), \tag{1}$$

where $P(s_0)$ is the prior probability of the initial state and $P(s_i|s_{i-1}, \mathcal{M}_t)$ denotes the probability of reaching a state s_i from a previous state s_{i-1} according to $\mathcal{M}_t$. This means that if s_i is not reachable from s_{i-1} executing a single action of A, then $P(s_i|s_{i-1}, \mathcal{M}_t) = 0$. Likewise if s_i is reachable then $P(s_i|s_{i-1}, \mathcal{M}_t)$ is given by the costs of actions applicable in s_{i-1}, according to $\mathcal{M}_t$, which we denote as $A(s_{i-1})$.

The sensing probability. Given a sensor model $\mathcal{M}_e = \langle Y, \Phi\rangle$ and a trajectory $\tau = \langle s_0, s_1, \ldots, s_n\rangle$, the *sensing probability* is the likelihood of perceiving an observation sequence $O = \langle o_1, o_2 \ldots, o_m\rangle$ with the sensor model:

$$P(O|\tau, \mathcal{M}_e) = \prod_{i=1}^{|\tau|} P(o_i|s_i, \mathcal{M}_e), \tag{2}$$

where $P(o_i|s_i, \mathcal{M}_e)$ denotes the probability of the assignment o_i for the observable variables Y given the actual state s_i. This probability is computed as $P(o_i|s_i, \mathcal{M}_e) = \prod_{Y_i \in Y} P(Y_i = v_i|s_i)$. For instance, going back to the example of the blocksworld with two hands, for a state $s = \{\mathsf{hold}_{\mathsf{h1,b1}}, \mathsf{hold}_{\mathsf{h2,b2}}\}$, we have that:

$$P(\mathsf{hold}_\mathsf{x} = \mathsf{b1} \wedge \mathsf{hold}_\mathsf{y} = \mathsf{b2}|s) = 0.42$$
$$P(\mathsf{hold}_\mathsf{x} = \mathsf{b1} \wedge \mathsf{hold}_\mathsf{y} = \epsilon|s) = 0.28$$
$$P(\mathsf{hold}_\mathsf{x} = \epsilon \wedge \mathsf{hold}_\mathsf{y} = \mathsf{b2}|s) = 0.18$$
$$P(\mathsf{hold}_\mathsf{x} = \epsilon \wedge \mathsf{hold}_\mathsf{y} = \epsilon|s) = 0.12$$

The observation decoding problem

Unlike the traditional formulation of observation decoding in HMMs (Ghahramani 2001), we define here an *observation decoding* task for an incomplete input sequence of observations;i.e., the number of observations of the sequence may be smaller than the number of states of the agent trajectory. This means that the set of possible hypothesis (the possible trajectories) may be infinite. Examples of this are domains with reversible actions like the *blocksworld* (Slaney and Thiébaux 2001). Further, we formulate the *observation decoding* task leveraging a factored representation, so exponential spaces of states and observations can be compactly encoded.

Definition 2 (The observation decoding task). *An observation decoding task is a triple $T = \langle \mathcal{M}_t, \mathcal{M}_e, O\rangle$ where $\mathcal{M}_t = \langle F, A\rangle$ is a planning model, $\mathcal{M}_e = \langle Y, \Phi\rangle$ is a sensor model, and $O = \langle o_0, o_1, \ldots, o_m\rangle$ is an input observation sequence.*

The solution to an observation decoding task T is the most likely state trajectory that generates O with regard to both input models, $\mathcal{M}_t$ and $\mathcal{M}_e$.

Definition 3 (The most likely trajectory). *Given some observation sequence O, a planning model $\mathcal{M}_t$, and a sensor model $\mathcal{M}_e$, the most likely trajectory τ^* is defined as*

$$\tau^* = \arg\max_{\tau \in \mathcal{T}} P(O, \tau|\mathcal{M}_t, \mathcal{M}_e), \tag{3}$$

where $\mathcal{T}$ is the set of trajectories that can be synthesized with a planning model $\mathcal{M}_t$ and $P(O, \tau|\mathcal{M}_t, \mathcal{M}_e) = P(\tau|\mathcal{M}_t) \cdot P(O|\tau, \mathcal{M}_e)$ is the joint likelihood of an observation sequence O and the trajectory τ (i.e., the probability that O and τ occur at the same time).

Example of observation decoding

Consider the grid domain of Figure 1, whose $\mathcal{M}_e$ was previously presented, and the observation sequence $O = \langle o_1, o_2, o_3 \rangle$, where $o_1 = \langle 3, 1 \rangle$, $o_2 = \langle 3, 2 \rangle$ and $o_3 = \langle 3, 5 \rangle$ are three observations of the variable $\mathsf{loc}_{x,y}$. The set F of $\mathcal{M}_t$ contains the fluents $\mathsf{loc}_{i,j}$ for each cell, and the set A is the four direction moves (N,S,W,E), all with cost=1.

The most likely trajectory τ that solves $T = \langle \mathcal{M}_t, \mathcal{M}_e, O \rangle$ is the one represented with a solid line in Figure 1, $\tau = \langle$N,W,N,N,N,E$\rangle$, which results from executing these six actions in the initial cell $(3, 1)$. The joint probability of τ and O for T, $P(\tau|\mathcal{M}_t) \cdot P(O|\tau, \mathcal{M}_e)$, is $0.225^2 \cdot 0.25^4 = 0.0001977539$ because two moves of τ reach an open tile ($0.25 \cdot 0.90$, where 0.25 is the transition probability and 0.9 is the emission probability from an open tile), and four moves of τ traverse a covered tile ($0.25 \cdot 1$, being 0.25 the transition probability and 1 the emission probability of ϵ, the empty sensor reading). No other trajectory consistent with T and higher probability can be found. For instance, the probability of the shorter trajectory corresponding to the four action sequence $\tau = \langle$N,N,N,N$\rangle$ (the dotted line in Figure 1), is $P(\tau|\mathcal{M}_t) \cdot P(O|\tau, \mathcal{M}_e) = 0.225^2 \cdot 0.025^2 = 0.00003164062 < 0.0001977539$ since the emission probability is 0.9 for the first and last move and 0.1 for the two middle moves ($P(\epsilon|\mathsf{loc}_{3,3}) = P(\epsilon|\mathsf{loc}_{3,4}) = 0.1$).

Observation decoding via classical planning

Computing Equation 3 is intractable as the set of possible trajectories $\mathcal{T}$ in some domains is potentially infinite. This section explains how to effectively compute τ^* in a single planning episode with an optimal planner.

Considering the definitions of the *synthesis* and the *sensing* probabilities (equations 1 and 2 respectively), we rewrite equation 3 as the following product:

$$\tau^* = \arg\max_{\tau \in \mathcal{T}} P(s_0) \prod_{i=1}^{|\tau|} P(s_i|s_{i-1}, \mathcal{M}_t) \cdot P(o_i|s_i, \mathcal{M}_e), \tag{4}$$

Classical planners are traditionally engineered to minimize the cost of the actions in the solution plan (aka `total-cost`). Further, classical planners deal with a fixed initial state instead of handing a probabilistic distribution over the set of possible initial states. With this regard, we define a classical planning compilation that transforms the probability maximization of Equation 4 into an equivalent cost minimization (Equation 5) by: (1), working on the log scale (Jiménez, Coles, and Smith 2006; Little and Thiebaux 2007) and (2), fixing each of the possible initial states:

$$\tau^* = \arg\min_{\tau \in \mathcal{T}} \sum_{i=1}^{|\tau|} -log(P(s_i|s_{i-1}, \mathcal{M}_t)) - log(P(o_i|s_i, \mathcal{M}_e)), \tag{5}$$

As in classical planning, Equation 5 assumes s_0 is known and so $P(s_0) = 1$, which does not affect the cost minimization. Considering different possible initial states with different prior probabilities is however straightforward: in such case $-log(P(s_0))$ is added to the cost minimization shown in Equation 5.

$\mathsf{pre}(\mathsf{transit_a})$	$pre(a) \cup \{mode_t\}$
$\mathsf{eff}(\mathsf{transit_a})$	$eff(a) \cup \{\neg mode_t, mode_e\}$
$\mathsf{pre}(\mathsf{sense_{o_i}})$	$\{reached_{i-1}, mode_e\}$
$\mathsf{eff}(\mathsf{sense_{o_i}})$	$\{\neg reached_{i-1}, reached_i\} \cup \{\neg mode_e, mode_t\}$

Table 1: *Transition actions* extend the original actions $a \in A$ while *sensing actions* process observation $o_i \in O$ according to the input sensor model. Their execution is interleaved.

Reducing observation decoding to planning

Given an *observation decoding* task $T = \langle \mathcal{M}_t, \mathcal{M}_e, O \rangle$ we build a classical planning problem $P' = \langle F', A', I', G' \rangle$ s.t. an optimal plan for P' induces a τ^* trajectory that minimizes Equation 5. Next we detail each element in the compiled classical planning problem:

- $F' = F \cup \{reached_i\}_1^{|O|} \cup \{disabled, mode_t, mode_e\}$ where:
  - F are the fluents of the input planning model $\mathcal{M}_t$.
  - $reached_i$ indicates that the observation $o_i \in O$ ($1 \leq i \leq |O|$) is processed.
  - $disabled$ is a flag indicating that an observation could not be processed.
  - $mode_t$ and $mode_e$ are flags that alternatively enable the application of *transition actions* and *sensing actions*.
- $I' = s_0$ is the fixed initial state.
- $G' = \{reached_{|O|}, \neg disabled\}$, the goal condition is that the full sequence of input observations is successfully processed.
- $A' = A_t \cup A_e$ where A_t are the *transition actions*, original actions in the input planning model $\mathcal{M}_t$ but extended with a cost value representing the logarithm of the corresponding *transition probability*. Actions in A_e are the *sensing actions*, new actions to process an observation according to the input sensing model $\mathcal{M}_e$. In more detail:
  - **Transition actions.** For each action $a \in A$ there is a $\mathsf{transit_a} \in A_t$ action. Table 1 shows that $\mathsf{transit_a}$ extends the precondition and effects of the corresponding original action to update the $\{mode_t, mode_e\}$ flags. Additionally, a $\mathsf{transit_a}$ action has a cost value representing the log probability of the state transition encoded by that action:

    $$cost(\mathsf{transit_a}) = -log \frac{cost(a)}{\sum_{a' \in A(s)} cost(a')},$$

    where $A(s) \subseteq A$ is the subset of applicable actions at the state where the action is applied.
  - **Sensing actions.** For every observation $o_i \in O$, A_e contains an action $sense_{o_i}$, Table 1 details how a $sense_{o_i}$ action is built. The set A_e includes also a special $sense_\epsilon$ action that does not require precondition

$reached_{i-1}$. This way $sense_\epsilon$ can always be applied in the *sensing mode* (when $mode_e$ holds), which allow us to deal with the fact that input sequence of observations may miss an unbound number of intermediate states. Each sensing action is associated with a set of state-dependent costs $-log(P(Y_i = v|s, \mathcal{M}_e))$ that represent the log probability of each observed $Y_i = v$, from the current state s, and according to the input sensing model $\mathcal{M}_e$. There are two special cases treated differently aiming efficiency: $P(Y_i = v|s, \mathcal{M}_e) = 1$, which is ignored because $log(1) = 0$, and $P(Y_i = v|s, \mathcal{M}_e) = 0$, which is compiled into a conditional effect that adds the $disabled$ fluent (which prevents G' to be achieved).

The size of the classical planning problem that results from the compilation depends on: (1), the number of functions in the Φ set that determines the number of state-dependent cost increments in a *sensing action* and (2), the size of the input observation sequence (i.e., $|O|$), that determines the number of sensing actions.

Theorem 4. *Completeness. Any trajectory τ^* that solves an observation decoding task $T = \langle \mathcal{M}_t, \mathcal{M}_e, O\rangle$ can be computed with a plan π^* that optimally solves the classical planning problem $P' = \langle F', A', I', G'\rangle$.*

. The only extra precondition added to the original actions is the Boolean flag $mode_t$ (to interleave the execution of a $transit_a \in A_t$ action with a $sense_{o_i} \in A_e$ action). On the other hand, sensing actions do not modify the original state variables since they only update the $mode_t$ and $mode_e$ flags (and the `total-cost` according to $\mathcal{M}_e$). This means that the compilation is not introducing further constraints for the application of the original actions. □

Theorem 5. *Soundness. An optimal solution plan π^* for the classical planning problem $P' = \langle F', A', I', G'\rangle$ induces a trajectory τ^* that solves the observation decoding task $T = \langle \mathcal{M}_t, \mathcal{M}_e, O\rangle$.*

. According to the definition of the compilation, the execution of a $\mathsf{transit_a}$ action produces a cost increment of the log probability of the state transition represented by this action. The compilation forces that after a transition is executed then a single sensing action is executed. According to the definition of the compilation, the execution of a *sensing action* produces a cost increment that is defined by the sum of the log probabilities of all the $(Y_i = v)$ values that are observed from the current state. This means that, once the full sequence of input observations is successfully processed (i.e. the goal condition G' that any plan π must hold for the classical planning problem P') we have that $cost(\pi) = -log(P(\tau|\mathcal{M}_t) \cdot P(O|\tau, \mathcal{M}_e))$. If the plan π is optimal, then the most likely trajectory is the sequence of states traversed by this plan. □

Illustrative example

In this section we compare the proposed approach to an HMM through an example. This comparison highlights some limitations of HMM that can be overcome through a planning-based approach to observation decoding.

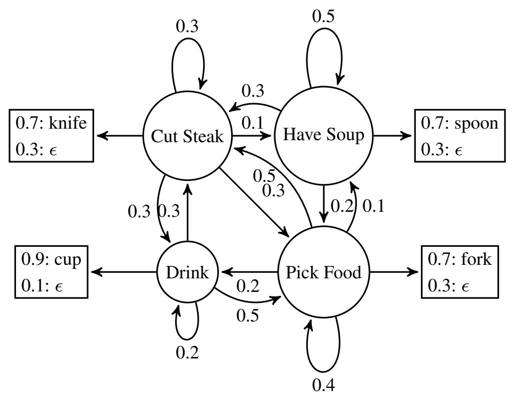

Figure 4: A four-state HMM for the activity recognition of *eating*.

The HMM for an eating activity displayed in Figure 4 is borrowed from (Kim, Helal, and Cook 2009), where we kept the same transition probabilities and modified the other two. The HMM has four hidden states, $\mathsf{CutSteak}$, Drink, $\mathsf{PickFood}$ and $\mathsf{HaveSoup}$, all represented with a single fluent, plus one state dummy that represents the initial state of the HMM. There is one observable variable $Y = \{\mathsf{Utensil}\}$, such that $D_{\mathsf{Utensil}} = \{\mathsf{spoon}, \mathsf{fork}, \mathsf{knife}, \mathsf{cup}, \epsilon\}$[1]. Therefore, the emission of an observation of the variable $\mathsf{Utensil}$ is a value within D_{Utensil}. The emission probabilities of the sensor model are shown in the figure; e.g. $\Phi_{\mathsf{Utensil}}(\mathsf{cup}, \mathsf{Drink}) = 0.9$, $\Phi_{\mathsf{Utensil}}(\epsilon, \mathsf{Drink}) = 0.1$. We will assume $P(\mathsf{Drink} = 0.5)$ and $P(\mathsf{HaveSoup} = 0.5)$.

This HMM can be represented by $\langle \mathcal{M}_t, \mathcal{M}_e\rangle$, and then be transformed to $P' = \langle F', A', I', G'\rangle$ via our formalism. Specifically, F' will comprise $F = \{\mathsf{CutSteak}, \mathsf{Drink}, \mathsf{PickFood}, \mathsf{HaveSoup}\}$ plus the necessary fluents for processing the input observations and for switching from transit to sense actions. $I' = \mathsf{dummy}$ is the *dummy state* of the HMM with transition probabilities $P(\mathsf{Drink}|I') = 0.5$ and $P(\mathsf{HaveSoup}|I') = 0.5$. G' includes a $reached$ fluent for each input observation in O, and A' is the set of compiled actions. Table 2 shows an action of A' that defines the transition from the state Drink to the state $\mathsf{PickFood}$, whose cost is $-log\ 0.5$.

Let's assume the actual trajectory of the agent is $\tau = \langle \mathsf{Drink}, \mathsf{PickFood}, \mathsf{HaveSoup}\rangle$. Following, we discuss the observation decoding for three input observation sequences.

Case 1. $O_1 = \langle \mathsf{cup}, \mathsf{fork}, \mathsf{spoon}\rangle$. In this case, we have an observation for every executed step of the agent and both the HMM and our formalism return the trajectory $\tau = \langle \mathsf{Drink}, \mathsf{PickFood}, \mathsf{HaveSoup}\rangle$. Table 3 shows the ground sensing action for observation $o_2 = \langle \mathsf{fork}\rangle$ in O_1.

Case 2. $O_2 = \langle \mathsf{cup}, \epsilon, \mathsf{spoon}\rangle$. In this case, the malfunctioning of the sensor emits an empty reading after having observed $o = \langle \mathsf{cup}\rangle$. Both $\tau_1 = \langle \mathsf{Drink}, \mathsf{PickFood}, \mathsf{HaveSoup}\rangle$ and $\tau_2 = \langle \mathsf{Drink}, \mathsf{CutSteak}, \mathsf{HaveSoup}\rangle$ can explain O_2 but

[1]We include ϵ in the variable domain because HMMs cannot represent missing data.

$\mathsf{pre}(\mathsf{transit}_{\mathsf{Drink-PickFood}})$	$\mathsf{Drink} \wedge mode_t$
$\mathsf{eff}(\mathsf{transit}_{\mathsf{Drink-PickFood}})$	$\neg\mathsf{Drink} \wedge \mathsf{PickFood} \wedge$ $\neg mode_t \wedge mode_s$

Table 2: Transit action for the HMM in Figure 4.

$\mathsf{pre}(\mathsf{sense}_{\mathsf{fork}_2})$	$reached_{\mathsf{cup}_1} \wedge mode_e$
$\mathsf{eff}(\mathsf{sense}_{\mathsf{fork}_2})$	$\neg reached_{\mathsf{cup}_1} \wedge reached_{\mathsf{fork}_2} \wedge$
	$\neg mode_e \wedge mode_t$
	when CutSteak $(disabled)$
	when HaveSoup $(disabled)$
	when Drink $(disabled)$
	when PickFood
	increase $total\_cost\ -log\ (0.7)$

Table 3: Sensing action for observation $o_2 = \langle \mathsf{fork} \rangle$ in input sequence O_1.

τ_1 is the trajectory with maximum joint likelihood and both approaches return τ_1.

Case 3. $O_3 = \langle \mathsf{cup}, \mathsf{spoon} \rangle$. An HMM requires the observation sequence and the trajectory to be synchronized, meaning that at least there should be one observation per state and, thus, it cannot handle this case. Our approach is able to bypass this limitation thanks to planning, as it can fill the gaps in the observation sequence. In other words, HMMs can only deal with missing observations if these are made explicit through a special token (ϵ) as in Case 2, while the presented approach can deal with true null observations (Case 3) since it assumes that there may be an unbounded number of missing observations.

Experimental evaluation

The goal of this section is to evaluate the effectiveness of using the sensor model for the observation decoding task. For that purpose, we compared the performance of observation decoding with sensor models (OD_S), and without sensor models (OD_N). OD_N follows the reduction-to-planning scheme presented in (Ramírez and Geffner 2010) but updated to support the emission of more than one variable per observation. It basically consists in finding the optimal plan that complies with an observation sequence O.

Despite supporting state-dependent action costs is trivial in state-based forward search, current off-the-shelf PDDL planners are not able to handle this functionality. This is due to the difficulty of computing informed goal-distance estimates with state-dependent action costs. Recent relevant work makes progress in this direction (Geißer, Keller, and Mattmüller 2015; 2016) but, unfortunately, this has not yet output a PDDL planner. In our case, planning problems were optimally solved with an A* search guided by the blind heuristic (h=0), using the search code of the METRIC-FF (Hoffmann 2003) planner.

The experiment consists in providing O to OD_S and OD_N and measuring the similarity of their trajectories with the actual trajectory that induced O. For generating O, we define a sensor model with Y observable variables which domain of values depend on a subset of the fluents of F. This is done by splitting Y into variables with high observability and with low observability (more details below).

The domains used in this experiment are the following:

- BLINDSPOTS: This is the domain used in Figure 1 where $|Y| = 1$. The only observable variable is the location of the agent, which is only visible from the white tiles. For this experiment, the grids were randomly generated as well as the initial and final positions of the agent.
- INTRUSION: This domain emulates attacks on host computers (Ramírez and Geffner 2010; Pereira, Oren, and Meneguzzi 2017). The attacker needs to perform a number of steps in order to steal data or vandalize a host. We defined 10 observable variables and added a few modifications to allow different ways to accomplish the goal so that some attack paths are more observable than others.
- BLOCKS 2H: The classic *blocksworld* domain with four operators but using an additional hand. We define several variables to emit the status of a block, and one variable for each hand, one of them highly observable.
- OFFICE: This is a transportation domain used in (Alford, Kuter, and Nau 2009), where a robot needs to pick up and deliver packages in a building. The building is represented as rooms connected by doors, and the robot may need to go through many rooms to deliver a package. In this domain, we define an observable variable for each package and make it so packages are easily traceable (high observability) in some rooms while in others not.

Table 4 shows the four domains. The values of column H (high observability) and column L (low observability) denote the degree of observability applied to the Y variables of each domain. As we can see, three different sensor models are defined for each domain. For instance, the sensor model 100-0 of the BLINDSPOTS defines full observability for the open tiles and null observability for the covered tiles.

We created a dataset composed of 50 trajectories for each domain, and the input observation sequences O were generated by applying the three sensor models of Table 4 to these trajectories. Therefore, each combination domain-model was tested over 50 observation sequences. This dataset and the source code for the experiments are available at https://github.com/anonsub/observation_decoding.

In order to compare the similarity between two trajectories or, equivalently, between two plans, we selected the δ_α version of the *diversity* metric presented in (Srivastava et al. 2007). This metric interprets a plan as a bag of actions and computes the similarity between two plans as the set-difference between the two. Formally:

$$\delta_\alpha(\pi_i, \pi_j) = \frac{|S_i - S_j|}{|S_i| + |S_j|} + \frac{|S_j - S_i|}{|S_i| + |S_j|}$$

where S_i and S_j are the bag of actions of plans π_i and π_j, respectively. A value of 0 for this metric represents complete similarity of plans while a value of 1 represents complete diversity. For this experiment we computed $\delta_\alpha(\pi, \pi_S)$ and $\delta_\alpha(\pi, \pi_N)$ where π is the plan that induced O, and π_S and

Domain	H	L	OD_S	OD_N
BLINDSPOTS	100	0	0.03	0.18
	80	20	0.08	0.20
	60	40	0.11	0.17
INTRUSION	100	0	0	0.58
	80	20	0.07	0.18
	60	40	0.13	0.14
BLOCKS 2H	100	0	0	0.34
	80	20	0.05	0.27
	60	40	0.07	0.26
OFFICE	100	0	0	0.58
	80	20	0.23	0.38
	60	40	0.16	0.23

Table 4: Comparison between OD_S and OD_N: domains, observability degree for H fluents, observability degree for L fluents, average $\delta_\alpha(\pi, \pi_S)$, average $\delta_\alpha(\pi, \pi_N)$.

π_N are the plans corresponding to the trajectories inferred by OD_S and OD_N, respectively.

The obtained results are shown in Table 4. The figures in columns OD_S and OD_N are the average values of $\delta_\alpha(\pi, \pi_S)$, and $\delta_\alpha(\pi, \pi_N)$, respectively, over the 50 samples. We can observe that the average diversity of OD_N is always higher than OD_S for every combination domain-model, meaning that OD_S decodes trajectories more accurately. It is also noticeable that the difference of diversity between both approaches narrows down as we get closer to a uniform distribution that sets the same degree of observability for all the domain objects. This seems reasonable since not considering the sensor model is equivalent to assuming a uniform distribution of observations where H=L.

It is also worth noting the significant diversity drop of OD_N from sensor model 100-0 to 80-20 in the INTRUSION domain. This happens because only two solution paths exist for these problems and increasing the low observability degree from 0 to 20 uncovers some variables that identify almost unambiguously the solution path.

In conclusion, exploiting the absence of observable information is as valuable as using the existing observations for predicting an agent trajectory, and this is what the OD_S model brings.

Beyond plan and goal recognition

Observation decoding with sensor models may be beneficial to planning-related recognition tasks. Our formalism can be plugged directly into a Goal Recognition problem (Ramírez and Geffner 2010) and other derived problems offering two main advantages: (1) the exploitation of a sensor model to build more accurate estimates about the acting agent, and (2) support for observations consisting of multiple observable variables.

The purpose of this section is to shed some light on the generalization of recognition tasks and provide a single task definition that allows recognizing any aspect of the acting agent, namely goals, initial states, intermediate states, plans or any combination of these. In order to present this new view of a recognition task, we need first to define the *likelihood of an observation sequence*:

$$P(O|\mathcal{M}_t, \mathcal{M}_e) = \sum_{\tau} P(O, \tau|\mathcal{M}_t, \mathcal{M}_e)$$
$$\approx \max_{\tau} P(O, \tau|\mathcal{M}_t, \mathcal{M}_e)$$

This likelihood is a *generating probability* as it defines the probability of generating O with the models $\mathcal{M}_t$ and $\mathcal{M}_e$. Its exact computation takes into account every τ that can emit O, but a common approximation is to assume that this probability is dominated by its largest term. Note that this approximation can be computed using the cost of the plan inferred by our observation decoding task.

A recognition task is about finding the most likely element among a set of hypothesis. For example, given a set of possible goals of an agent, Goal Recognition seeks to identify the real goal the agent is pursuing. In our notion of recognition task, we define the set of hypothesis $\mathcal{H}$ as a set of *incomplete trajectories* identical in shape to an observation sequence.

Definition 6 (The recognition task). *A recognition task is a triple $T = \langle \mathcal{M}_t, \mathcal{M}_e, \mathcal{H} \rangle$ where $\mathcal{M}_t = \langle F, A \rangle$ is a planning model, $\mathcal{M}_e = \langle Y, \Phi \rangle$ is a sensor model, and $\mathcal{H}$ is a set of hypothesis.*

The solution to a recognition task T is the hypothesis $H \in \mathcal{H}$ with maximum likelihood.

$$H^* = \arg\max_{H \in \mathcal{H}} P(H|\mathcal{M}_t, \mathcal{M}_e) \qquad (6)$$

For instance, the goal recognition task solved in (Ramírez and Geffner 2010), where the input data are the initial state I, a plan observation $\langle o_1, \ldots, o_n \rangle$, and a set of goals $\mathcal{G}$, amounts to finding the most likely hypothesis among the set $\mathcal{H} = \{\langle I, o_1, \ldots, o_n, G \rangle | G \in \mathcal{G}\}$ ($\mathcal{H}$ are incomplete trajectories that only differ in G). In other words, we solve $|\mathcal{H}|$ *observation decoding* tasks to find the last observation (G) that makes an observation sequence H more likely.

Our definition of *recognition task* broadens the application of the reduction-to-planning approach to a wider range of features of the acting agents. This view allows us to define an arbitrary number of hypothesis where the differences may lie in any state or action.

Conclusions

We have formulated a probabilistic sensor model to allow an observer infer helpful information about the acting agent from the partial observations emitted by sensors. The results show that the agent trajectory is more accurately computed when the sensor model is taken into account. The lesson learned from this work is that important gains are obtainable when sensor models exploit the non-observable data as well. Furthermore, the joint likelihood underpinning the observation decoding task lays the foundations for defining a generalized recognition task able to identify any feature of the agent behaviour (namely goals, initial states, intermediate states, plans or any combination of these).

Acknowledgments

This work is supported by the Spanish MINECO project TIN2017-88476-C2-1-R. D. Aineto is partially supported by the *FPU16/03184* and S. Jiménez by the *RYC15/18009*.

References

Aineto, D.; Jiménez, S.; Onaindia, E.; and Ramírez, M. 2019. Model Recognition as Planning. In *30th Int. Conf. on Automated Planning and Scheduling, ICAPS*, 13–21.

Aineto, D.; Jiménez, S.; and Onaindia, E. 2019. Learning action models with minimal observability. *Artificial Intelligence Journal* 275:104–137.

Alford, R. W.; Kuter, U.; and Nau, D. 2009. Translating HTNs to PDDL: A Small Amount of Domain Knowledge Can Go a Long Way. In *21st International Joint Conference on Artificial Intelligence, IJCAI*, 1629–1634.

Amir, E., and Chang, A. 2008. Learning partially observable deterministic action models. *Journal of Artificial Intelligence Research* 33:349–402.

Bonet, B.; Palacios, H.; and Geffner, H. 2009. Automatic Derivation of Memoryless Policies and Finite-State Controllers Using Classical Planners. In *19th International Conference on Automated Planning and Scheduling, ICAPS*.

Bonet, B.; Palacios, H.; and Geffner, H. 2010. Automatic derivation of finite-state machines for behavior control. In *24th AAAI Conference on Artificial Intelligence*.

Cresswell, S. N.; McCluskey, T. L.; and West, M. M. 2013. Acquiring planning domain models using LOCM. *The Knowledge Engineering Review* 28(02):195–213.

Freedman, R. G.; Jung, H.; and Zilberstein, S. 2014. Plan and activity recognition from a topic modeling perspective. In *24th International Conference on Automated Planning and Scheduling, ICAPS*.

Geißer, F.; Keller, T.; and Mattmüller, R. 2015. Delete relaxations for planning with state-dependent action costs. In *24th International Joint Conference on Artificial Intelligence, IJCAI*, 1573–1579.

Geißer, F.; Keller, T.; and Mattmüller, R. 2016. Abstractions for planning with state-dependent action costs. In *26th International Conference on Automated Planning and Scheduling, ICAPS*, 140–148.

Ghahramani, Z. 2001. An Introduction to Hidden Markov Models and Bayesian Networks. *Journal of Pattern Recognition and Artificial Intelligence* 15(1):9–42.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *J. Artif. Intell. Res.* 20:291–341.

Jiménez, S.; Coles, A.; and Smith, A. 2006. Planning in probabilistic domains using a deterministic numeric planner. In *25th Workshop of the UK Planning and Scheduling Special Interest Group*.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence* 101(1-2):99–134.

Keren, S.; Gal, A.; and Karpas, E. 2014. Goal recognition design. In *24th International Conference on Automated Planning and Scheduling, ICAPS*, 154–162.

Keren, S.; Gal, A.; and Karpas, E. 2019. Goal Recognition Design in Deterministic Environments. *Journal of Artificial Intelligence Research* 65:209–269.

Kim, E.; Helal, S.; and Cook, D. 2009. Human Activity Recognition and Pattern Discovery. *IEEE Pervasive Computing* 9(1):48–53.

Kucera, J., and Barták, R. 2018. LOUGA: learning planning operators using genetic algorithms. In *Pacific Rim Knowledge Acquisition Workshop, PKAW-18*, 124–138.

Kulkarni, A.; Srivastava, S.; and Kambhampati, S. 2019. A unified framework for planning in adversarial and cooperative environments. In *AAAI Conference on AI*, 2479–2487.

Little, I., and Thiebaux, S. 2007. Probabilistic planning vs. replanning. In *ICAPS Workshop on IPC: Past, Present and Future*.

Nazerfard, E., and Cook, D. J. 2015. CRAFFT: an activity prediction model based on bayesian networks. *J. Ambient Intelligence and Humanized Computing* 6(2):193–205.

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2017. Landmark-based heuristics for goal recognition. In *31st AAAI Conference on Artificial Intelligence*. AAAI Press.

Pozanco, A.; E.-Martín, Y.; Fernández, S.; and Borrajo, D. 2018. Counterplanning using goal recognition and landmarks. In *27th International Joint Conference on Artificial Intelligence, IJCAI*, 4808–4814.

Ramírez, M., and Geffner, H. 2009. Plan Recognition as Planning. In *21th International Joint conference on Artifical Intelligence, IJCAI*, 1778–1783. AAAI Press.

Ramírez, M., and Geffner, H. 2010. Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. In *24th AAAI National Conference on Artificial Intelligence*.

Ramírez, M., and Geffner, H. 2011. Goal recognition over pomdps: Inferring the intention of a POMDP agent. In *22nd International Joint Conference on Artificial Intelligence, IJCAI*, 2009–2014.

Silver, D., and Veness, J. 2010. Monte-Carlo planning in large POMDPs. In *Advances in neural information processing systems*, 2164–2172.

Slaney, J., and Thiébaux, S. 2001. Blocks world revisited. *Artificial Intelligence* 125(1-2):119–153.

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan recognition as planning revisited. In *25th International Joint Conference on Artificial Intelligence, IJCAI*, 3258–3264.

Srivastava, B.; Nguyen, T. A.; Gerevini, A.; Kambhampati, S.; Do, M. B.; and Serina, I. 2007. Domain independent approaches for finding diverse plans. In *20th Int. Joint Conference on Artificial Intelligence, IJCAI*, 2016–2022.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence* 171(2-3):107–143.

Yu, S. 2016. *Hidden Semi-Markov Models: Theory, Algorithms and Applications*. Boston: Elsevier.