Goal Recognition over POMDPs: Inferring the Intention of a POMDP Agent*

Miquel Ramírez
Universitat Pompeu Fabra
08018 Barcelona, SPAIN
miquel.ramirez@upf.edu

Hector Geffner
ICREA & Universitat Pompeu Fabra
08018 Barcelona, SPAIN
hector.geffner@upf.edu

Abstract

Plan recognition is the problem of inferring the goals and plans of an agent from partial observations of her behavior. Recently, it has been shown that the problem can be formulated and solved using planners, reducing plan recognition to plan generation. In this work, we extend this model-based approach to plan recognition to the POMDP setting, where actions are stochastic and states are partially observable. The task is to infer a probability distribution over the possible goals of an agent whose behavior results from a POMDP model. The POMDP model is shared between agent and observer except for the true goal of the agent that is hidden to the observer. The observations are action sequences O that may contain gaps as some or even most of the actions done by the agent may not be observed. We show that the posterior goal distribution $P(G|O)$ can be computed from the value function $V_G(b)$ over beliefs b generated by the POMDP planner for each possible goal G. Some extensions of the basic framework are discussed, and a number of experiments are reported.

Introduction

Plan recognition is the problem of inferring the goals and plans of an agent from partial observations of her behavior (Cohen, Perrault, and Allen 1981; Pentney et al. 2006; Yang 2009). The problem arises in a number of applications, and has been addressed using a variety of methods, including specialized procedures (Kautz and Allen 1986; Avrahami-Zilberbrand and Kaminka 2005), parsing algorithms (Pynadath and Wellman 2002; Geib and Goldman 2009) and Bayesian networks inference procedures (Bui 2003). In almost all cases, the space of possible plans or activities to be recognized is assumed to be given by a suitable library of policies or plans.

Recently, two formulations have approached the plan recognition problem from a different perspective that replaces the need for a set of possible agent policies or plans, by an agent action model and a set of possible goals. The model expresses how the agent can go about achieving these goals and is used to interpret the observations. The result is a posterior probability distribution over the possible goals. In these approaches, the possible agent behaviors are encoded implicitly in the set of goals and action models, rather than explicitly as a library of plans. The advantage of these approaches to plan recognition is that they can leverage on model-based behavior generators; namely, planners. In (Ramirez and Geffner 2009; 2010), the model is classical planning model, namely, the initial state is fully known to agent and observer, and the actions have deterministic effects, while in (Baker, Saxe, and Tenenbaum 2009), the model is a Markov Decision Process (MDP), so that the states are fully observable, and actions have stochastic effects.

In this work, we extend the model-based approach to plan recognition over POMDP settings, where actions are stochastic and states are partially observable. The task is to infer a probability distribution over the possible goals of an agent whose behavior results from a POMDP (Partially Observable MDP) model. The model is shared between agent and observer except for the true goal of the agent that is hidden to the observer. The observations are action sequences that may contain gaps as some or even most of the actions done by the agent are not observed. We show that the posterior goal distribution can be computed from the value function over beliefs generated by a POMDP planner for each possible goal G. More precisely, executions are sampled from this value function, assuming that the agent tends to select the actions that look best, and the likelihood of the observations O given the goal G is approximated from these samples. In analogy to the other cases, the goal recognition problem over POMDPs is solved using an off-the-shelf POMDP planner. While POMDP planners do not scale up as well as MDP planners, and certainly much worse than classical planners, we show that still a rich variety of recognition problems involving incomplete information can be effectively modeled and solved in this manner. The expressive power and computational feasibility of the approach will be illustrated through a number of experiments over several domains.

The paper is organized as follows. We start with an example (Section 2), and then review previous approaches (Section 3), and POMDPs (Section 4). We then consider a preliminary formulation of POMDP goal recognition that assumes that all agent actions and observations are visible to

*Paper appearing also at Proc. IJCAI–11.

the observer (Section 5), and a more general form that assumes neither (Section 6). We then test the latter over several problems (Section 7) and summarize the contributions (Section 8).

Motivation

As an illustration of how a goal recognition problem can be naturally cast in the POMDP setting, consider an agent that is looking for an item A or B each of which can be in one of three drawers 1, 2, or 3, with probabilities $P(A@i)$ and $P(B@i)$ equal to:

$$P(A@1) = 0.6\ , P(A@2) = 0.4\ , P(A@3) = 0$$
$$P(B@1) = 0.1\ , P(B@2) = 0.6\ , P(B@3) = 0.3$$

The actions available to the agent are to open and close the drawers, to look for an item inside an open drawer, and to grab an item from a drawer if it's known to be there. Let us assume that the agent is a male, and hence that the probability that he doesn't observe the object in the drawer when the object is actually there is non-zero, say 0.2, but that the probability that he observes an object when it's not there is 0 indeed.

Let us assume that the possible goals G_1, G_2, and G_3 of the agent are to have item A, item B, or both, with priors 0.4, 0.4, and 0.2. We want to find out the goal posterior probabilities when the behavior of the agent is partially observed. In our setting, the observer gets to see some of the actions done by the agent, but not necessarily all of them. The observer must then fill up the gaps. Let us assume that it is observed that the agent opens drawer 1, then drawer 2, and then drawer 1 again; i.e.,

$$O = \{open(1), open(2), open(1)\}.$$

The most likely explanation of this observation trace is that the agent is looking for item A; else it wouldn't have started by looking in drawer 1 where the probability of finding B is 0.1. Then, it's likely that the agent didn't observe A in that drawer, that it closed it, and then looked for A in drawer 2. Then, probably the agent didn't find A in drawer 2, and thus looked again in drawer 1.

Indeed, the algorithm that we will describe, concludes that the posterior probabilities for the three possible goals are $P(G_1|O) = 0.6$, $P(G_2|O) = 0.1$, and $P(G_3|O) = 0.3$, with G_1 as the most likely goal.

Previous Approaches

As mentioned in the introduction, the problem of plan, goal, or activity recognition has been addressed in many ways, in most cases assuming that there is a library of possible plans or policies that represents the possible agent behaviors. The problem has been formulated in a variety of ways, as a deductive problem over a suitable logical theory (Kautz and Allen 1986), a matching problem over a suitable AND/OR graph (Avrahami-Zilberbrand and Kaminka 2005), a parsing problem over a grammar (Pynadath and Wellman 2002; Geib and Goldman 2009), and an inference task over a dynamic Bayesian network (Bui 2003).

Some recent approaches, however, attempt to map the *plan recognition problem* into *plan generation* to leverage on the performance of state-of-the-art planners. Ramirez and Geffner (2010) consider the problem of plan recognition over classical planning models where the goal of the agent is hidden to the observer. They show that the posterior distribution $P(G|O)$ over the possible agent goals G given a sequence of observations O, can be defined from the costs $c(G, O)$ of the plans that achieve G while complying with the observations O, and the costs $c(G, \overline{O})$ of the plans that achieve G while not complying with O. Indeed, they define the likelihood $P(O|G)$ as a monotonic (sigmoid) function of the difference in costs $c(G, \overline{O}) - c(G, O)$. Thus, when the best plans for G all comply with O, this difference will be positive, and the larger the difference, the larger the likelihood $P(O|G)$. In order to compute the posterior probabilities $P(G|O)$ for a set $\mathcal{G}$ of possible goals G, the likelihoods $P(O|G)$ are then derived in $2|\mathcal{G}|$ planner calls, and they are then plugged into Bayes rule along with the priors $P(G)$ to yield the posterior probabilities $P(G|O)$.

The other recent model-based approach to plan recognition is in the MDP setting where actions are assumed to have stochastic effects and states are fully observable (Baker, Saxe, and Tenenbaum 2009). Baker *et. al.* show that from the value function $V_G(s)$ that captures the expected cost from state s to the goal G, for every state s and goal G, it is possible to define the probability that the agent will take a given action a in s if her goal is G. From this probability $P(a|s, G)$ and simple manipulations involving basic probability laws, they derive the likelihood $P(O|s, G)$ that the agent performs a sequence of actions O given that she starts in s and pursues the goal G. As before, from the likelihoods and the goal priors $P(G)$, they derive the posterior probabilities $P(G|O)$ using Bayes rule. Once again, the main computational work is done by the planner, in this case an MDP planner, that must furnish the value function $V_G(s)$ for all goals G and states s. Notice that in both the classical and MDP formulations, *probabilities* are inferred from *costs*; in the first case, the costs $c(G, O)$ and $c(G, \overline{O})$, in the second, the expected costs $V_G(s)$. The formulation that we develop below takes elements from both formulations while extending them to the POMDP setting where actions are stochastic and states are just *partially observable*.

Background: Goal MDPs and POMDPs

Shortest-path MDPs provide a generalization of the state models traditionally used in heuristic search and planning in AI, accommodating stochastic actions and full state observability (Bertsekas 1995). They are given by

- a non-empty state space S,
- a non-empty set of goal states $S_G \subseteq S$,
- a set of actions A,
- probabilities $P_a(s'|s)$ for $a \in A$, $s, s' \in S$, and
- costs $c(a, s)$ for $a \in A$ and $s \in S$.

The goal states t are assumed to be absorbing and cost-free; meaning $P_a(t|t) = 1$ and $c(a, t) = 0$ for all $a \in A$. Goal MDPs are shortest-path MDPs with a known initial state s_0

and *positive action costs* $c(a, s)$ for all a and non-terminal states s. Shortest-path and Goal MDPs appear to be less expressive than *discounted reward* MDPs, where there is no goal, rewards can be positive, negative, or zero, and a parameter γ, $0 < \gamma < 1$, is used to discount future rewards. Yet, the opposite is true: discounted reward MDPs can be transformed into equivalent Goal MDPs, but the opposite transformation is not possible (Bertsekas 1995). The same holds for discounted reward POMDPs and Goal POMDPs (Bonet and Geffner 2009).

The solution to MDPs are functions π mapping states into actions. The expression $V^\pi(s)$ denotes the *expected cost* that results from following the policy π from the state s to a goal state, and it can be computed by solving a system of $|S|$ linear equations. The optimal policies are well-defined if the goal is reachable from every state, and corresponds to the policies π^* that minimize $V^\pi(s)$ over all states s. The optimal cost function $V^*(s) = V^\pi(s)$ for $\pi = \pi^*$, turns out to be the unique solution to the Bellman equation:

$$V(s) = \min_{a \in A} \left\{ c(a, s) + \sum_{s' \in S} P_a(s'|s) V(s') \right\} \quad (1)$$

for all $s \in S \setminus S_G$, and $V(s) = 0$ for $s \in S_G$. The Bellman equation can be solved by the *Value Iteration* (VI) method, where a value function V, initialized arbitrarily over non-goal states, is updated iteratively until convergence using the right-hand side of (1). The optimal policy π^* is the policy π_V that is *greedy* in the value function V

$$\pi_V(s) = \operatorname*{argmin}_{a \in A} \left\{ c(a, s) + \sum_{s' \in S} P_a(s'|s) V(s') \right\}. \quad (2)$$

when $V = V^*$. Recent variants of value iteration aim to exploit the use of lower bound (admissible) cost or heuristic functions to make the updates more focused and to achieve convergence on the states that are relevant only. One of the first such methods is *Real-Time Dynamic Programming* (RTDP), that in each trial simulates the greedy policy π_V, updating the value function V over the states that are visited (Barto, Bradtke, and Singh 1995). With a good initial lower bound V, RTDP and other recent heuristic search algorithms for MDPs, can deliver an optimal policy without even considering many of the states in the problem.

POMDPs (Partially Observable MDPs) generalize MDPs by modeling agents that have incomplete state information (Kaelbling, Littman, and Cassandra 1999) in the form of a prior belief b_0 that expresses a probability distribution over S, and a sensor model made up of a set of observation tokens Obs and probabilities $Q_a(o|s)$ of observing $o \in Obs$ upon entering state s after doing a. Formally, a *Goal POMDP* is a tuple given by:

- a non-empty state space S,
- an initial belief state b_0,
- a non-empty set of goal states $S_G \subseteq S$,
- a set of actions A,
- probabilities $P_a(s'|s)$ for $a \in A$, $s, s' \in S$,
- *positive* costs $c(a, s)$ for non-target states $s \in S$,
- a set of observations Obs, and
- probabilities $Q_a(o|s)$ for $a \in A, o \in Obs$, $s \in S$.

It is also assumed that goal states t are cost-free, absorbing, and fully observable; i.e., $c(a, t) = 0$, $P_a(t|t) = 1$, and $t \in Obs$, so that $Q_a(t|s)$ is 1 if $s = t$ and 0 otherwise. The *target beliefs* or goals are the beliefs b such that $b(s) = 0$ for $s \in S \setminus S_G$.

The most common way to solve POMDPs is by formulating them as completely observable MDPs over the *belief states* of the agent. Indeed, while the effects of actions on states cannot be predicted, the effects of actions on *belief states* can. More precisely, the belief b_a that results from doing action a in the belief b, and the belief b_a^o that results from observing o after doing a in b, are:

$$b_a(s) = \sum_{s' \in S} P_a(s|s') b(s'), \quad (3)$$

$$b_a(o) = \sum_{s \in S} Q_a(o|s) b_a(s), \quad (4)$$

$$b_a^o(s) = Q_a(o|s) b_a(s) / b_a(o) \quad \text{if } b_a(o) \neq 0. \quad (5)$$

As a result, the *partially observable* problem of going from an initial state to a goal state is transformed into the *completely observable* problem of going from one *initial belief state* into a *target belief state*. The Bellman equation for the resulting *belief* MDP is

$$V(b) = \min_{a \in A} \left\{ c(a, b) + \sum_{o \in Obs} b_a(o) V(b_a^o) \right\} \quad (6)$$

for non-target beliefs b and $V^*(b_t) = 0$ otherwise, where $c(a, b)$ is the expected cost $\sum_{s \in S} c(a, s) b(s)$.

Many of the methods used for solving POMDPs are MDP methods extended to deal with the infinite and dense set of possible belief states. In our experiments, we use RTDP-Bel (Bonet and Geffner 2000; 2009), which is a straightforward adaptation of RTDP to Goal POMDPs where states are replaced by belief states updated according to (6).

Goal Recognition: Complete Observations

Our first formulation of goal recognition over POMDPs is a direct generalization of the MDP account (Baker, Saxe, and Tenenbaum 2009). This account makes *two assumptions*. First, that the observation sequence $O = a_1, \ldots, a_n$ is *complete*, meaning that O contains *all the actions done by the agent up until* a_n, and hence, that there are no gaps in the sequence. Second, that *the states of the MDP are fully observable not only to the agent, but also to the observer.* The assumptions are pretty restrictive but serve to reduce the goal recognition problem to a simple probabilistic inference problem. In the POMDP setting, the second assumption translates into the (partial) observations gathered by the agent being visible also to the observer. Thus, in this setting, the observer gets two types of information: the *complete* sequence of actions O done by the agent, and the corresponding sequence of POMDP observation tokens $o \in Obs$ that the agent received. In the next section, we relax these two assumptions.

The POMDP is assumed to be known by both the agent and the observer, except for the actual goal G of the agent.

Instead, the set $\mathcal{G}$ of possible goals is given along with the priors $P(G)$. The posterior goal probabilities $P(G|O)$ can be obtained from Bayes rule:

$$P(G|O) = \alpha P(O|G)P(G) \tag{7}$$

where α is a normalizing constant that doesn't depend on G. The problem of inferring the posteriors $P(G|O)$ gets thus mapped into the problem of defining and computing the likelihoods $P(O|G)$. The key assumption is that if the agent is pursing goal G, the probability $P(a|b, G)$ that she will choose action a in the belief state b is given by the Boltzmann policy:

$$P(a|b, G) = \alpha' exp\{\beta\, Q_G(a, b)\} \tag{8}$$

where α' is a normalizing constant and β captures a 'soft rationality' assumption (Baker, Saxe, and Tenenbaum 2009): for large β, the agent acts greedily on Q_G (optimally if Q_G is optimal); for low β, the agent selects actions almost randomly.

The term $Q_G(a, b)$ expresses the expected cost to reach the goal G from b starting with the action a; i.e.,

$$Q_G(a, b) = c(a, b) + \sum_{o \in O} b_a(o) V_G(b^o_a) \tag{9}$$

where V_G is the value function for the POMDP assuming that the goal states are those in which G is true, $c(a, b)$ is the expected cost of action a in b, and $b_a(o)$ and b^o_a as defined above, stand for the probability that agent observes o after doing action a in b, and the probability distribution that results from doing a in b and actually observing o. The likelihood $P(O_i|b, G)$ of the observation sequence $O_i = a_i, \ldots, a_n$ given the belief b and the goal G, can be computed recursively as:

$$P(O_i|b, G) = \begin{cases} P(a_n|b, G) & \text{if } i = n, \text{ else} \\ P(a_i|b, G) \sum_o P(O_{i+1}|b^o_a, G)\, b_a(o)\,. \end{cases} \tag{10}$$

The likelihood $P(O|b, G)$ is then $P(O_i|b, G)$ for $i = 0$, which can be computed from the recursion and plugged into Bayes rule (7) to obtain the desired posterior goal probabilities $P(G|O)$. The POMDP planner enters into this formulation by providing the expected costs $V_G(b)$ to reach G from b, that are used via the factors $Q_G(a, b)$ for defining the probability that the agent will do the action a when in the belief state b (Eq. 8).

Goal Recognition: Incomplete Observations

In the account above, the information available to the observer contains both the sequence of actions O done by the agent, and the observations $o \in Obs$ that the agent receives from the environment. Moreover, the sequence of actions is assumed to be complete, so that all the agent actions are observed. In the account below, the sequence of actions O obtained by the observer may be incomplete and the observations received by the agent are not available.

As before, we assume a shared POMDP between agent and observer, except for agent goal G that belongs to the set $\mathcal{G}$ of possible goals but is hidden to the observer. Since the observation sequence $O = a_1, \ldots, a_n$ is not necessarily complete, we cannot assume that action a_{i+1} in O is the action that the agent did right after a_i. Yet, the posterior goal probabilities $P(G|O)$ can be derived using Bayes rule (7) from the priors $P(G)$ and the likelihoods $P(O|G)$ that can now be defined as

$$P(O|G) = \sum_\tau P(O|\tau)P(\tau|G) \tag{11}$$

where τ ranges over the possible executions of the agent given that she is pursuing goal G. Executions τ contain the complete sequence of agent actions.

We will say that *an execution τ complies with the observation sequence O* if the sequence O is embedded in the sequence τ. Defining then the probabilities $P(O|\tau)$ to 1 or 0 according to whether the execution τ complies with O or not, the sum in (11) can be approximated by *sampling* as

$$P(O|G) \approx m_0/m \tag{12}$$

where m is the total number of executions sampled for each goal G, and m_O is the number of such executions that comply with O.

For this approximation to work, executions τ for the goal G need to be sampled with probability $P(\tau|G)$. This can be accomplished by making the agent select the action a in a belief b with a probability $P(a|b, G)$ that results from the Boltzmann policy (8). As before, it is assumed that the POMDP planner, furnishes the value function $V_G(b)$ that encodes the expected cost from b to the goal.

Once the action a is sampled with probability $P(a|b, G)$, the resulting observation o, that is no longer assumed to be available to the observer, is sampled with probability $b_a(o)$. The resulting full traces $b_0, a_0, o_0, b_1, a_1, o_1, \ldots$ until the goal is reached are such that $b_{i+1} = b^o_a$ for $b = b_i$, $a = a_i$, and $o = o_i$, where a_i is sampled with probability $P(a_i|b_i, G)$, and o_i is sampled with probability $b_a(o_i)$ for $b = b_i$ and $a = a_i$.

The likelihoods $P(O|G)$ approximated through (11) are then plugged into Bayes rule (7) from which the posterior goal probabilities $P(G|O)$ are obtained.

The key computational burden in this account results from the calculation of the value function V_G over beliefs, that must be provided as before by the POMDP planner, and the simulated executions that need to be done for estimating the likelihoods $P(O|G)$ following (12).

Experiments

To evaluate the effectiveness of the goal recognizer described in Section 6, we used the POMDP solver GPT (Bonet and Geffner 2001) built around the RTDP-BEL algorithm. GPT supports a very expressive language to define POMDPs which have allowed us to test our approach over three challenging domains detailed next. We needed to make two modifications on the original GPT sources. We changed the code to have it built with the latest versions of the GNU C++ compiler and libraries, and added a method to simulate the policies computed by GPT. The software for the goal recognition part was implemented on PYTHON, making calls to GPT when necessary.

Name	$\|S\|$	$\|A\|$	$\|Obs\|$	$\|b_0\|$	$\|\mathcal{G}\|$	T
OFFICE	2,304	23	15	4	3	3.4
DRAWERS	3,072	16	16	6	3	4.5
KITCHEN	69,120	29	32	16	5	10.1

Table 1: Domains used in the evaluation. $|S|$ denotes number of states, $|A|$ number of actions, $|Obs|$ number of observations, $|b_0|$ cardinality of initial belief, $|\mathcal{G}|$ number of possible goals and T average time (in seconds) to compute $V_G(b_0)$ for each of the goals G.

Table 1 shows the number of states, actions, observations, goals, and possible initial states for each of the three domains we used to evaluate the proposed goal recognition scheme with incomplete observations. These are non–trivial POMDPs that feature uncertainty in the initial state, stochastic actions and stochastic sensors. We will describe the domains next.

The DRAWERS domain is the task described early in the paper. The agent goals are to be holding either an object named *A*, an object named *B* or both. The two objects are distributed in the drawers according to the probability distribution described in Section 2. The agent can open drawers, look for a particular object inside an open drawer and pick an object. All actions have unitary cost. There is a small chance – 30 out of every 100 times – that she does not find the object she is looking for, even if the object actually is in the inspected drawer. Singleton goals are considered to have equal prior probability (0.4). For the joint goal we set its prior probability to the product of the singleton goals. As an illustration, if the agent is observed to "look into drawer #3" then it is more likely the goal "hold B", since A can't be initially placed in drawer #3.

In the OFFICE domain, adapted from (Bui 2003), the agent being observed is a researcher who arrives early to her lab, which consists of two rooms: one is her office, where she has her workstation and a cabinet to store her coffee cup and blank paper. The other is the club, where the lab coffee machine and printer are placed. The two rooms are connected by a corridor. The researcher is initially on the corridor. The printer can start either out of paper, clogged, both or none. The agent goals are either to print an article, have a cup of coffee or both. To print an article, the researcher needs to get to her workstation, send the file to the printer queue and get to the printer to retrieve it. There is a small chance – 20 out of every 100 times – that the printer will get clogged while trying to print the article. If the printer is out of paper, the file is kept on the printer queue. In this case, the agent will need to fetch blank paper from the cabinet in her office. When the printer is clogged, the agent will have to execute several actions to service it. To have coffee, the agent needs to get the cup from the cabinet in her office and then walk to the coffee machine. As in the DRAWERS domain, all actions have a cost of 1, singleton goals have equal prior probability, and the joint goal prior probability is the product. For example, if the agent is observed to execute the actions "walk to workstation, walk from corridor into club" then the most likely goal will be "print article", since the agent doesn't need to get to the workstation if she's pursuing the goal "have coffee".

In the KITCHEN domain the agent is trying to cook one out of five possible dishes. There are ingredients $i1$, $i2$, $i3$, $i4$ which are placed at random on two cupboards. Each dish requires up to three different ingredients which are required to be mixed in a bowl. The agent can inspect the cupboards the find the ingredients it needs, having to move first in front of the cupboard of interest. Additionally the agent needs to get hold of three different objects – a tray, a pan and a pot – which are all located in a third cupboard. Whenever a recipe involves boiling or frying an ingredient, or a mix of them, the agent needs to place the required objects on the stove. The agent can navigate freely between the locations of the bowl, the stove or the cupboards and can carry as many ingredients as she sees fit. All goals have the same prior probability and action costs are uniform. Thus if the agent is observed to "take pan, take $i2$" then the most likely goals will be those dishes which require to fry a mix of ingredients that includes $i2$.

The synthetic dataset was built as follows. For each domain and goal, we first computed the value function V_G for each possible goal G using GPT. Then, we sampled 100 executions of the greedy policy based on V_G for each possible goal $G \in \mathcal{G}$. From these 100 sampled executions, 10 observation sequences O were obtained by sampling randomly 30%, 50% or 70% of the actions over an also randomly selected execution.

In the table and figures, we don't report the posterior goal distributions $P(G|O)$, but just the resulting *binary classifier* that maps the observation sequences O into sets of *most likely* goals G. These are the goals G that maximize the posterior probability $P(G|O)$. We denote this binary goal recognizer as $\text{GR}(m, \beta)$, where m and β stand for the two parameters of the algorithm: the number of samples for each goal used in the approximation of $P(O|G)$, and the β coefficient that expresses the level of noise in the action selection. The set of most likely goals are those G' that verify $P(G') = \max_{G \in \mathcal{G}} P(G|O)P(G)$ [1].

The classification instances are the pairs $\langle O, G \rangle$ over all the observation sequences O and goals G. An instance is positive (P) if O was generated with G, and negative (N) otherwise. A true positive (TP) is a positive instance classified as positive, while a false negative, is a negative instance classified as positive. True negatives (TN) and false negatives (FN) are defined in a similar manner. The numbers of instances in these different classes provide the standard measures for evaluating the quality of a classifier. In particular, TPR, FPR, ACC, and PPV measure the True Positive Rate, False Positive Rate, Accuracy, and Precision of a classifier, defined as TP/P, FP/N, $TP + TN/P + N$, and $TP/TP + FP$ respectively.

Figure 1 shows the aggregate results of the goal recognizer $\text{GR}(m, \beta)$ over all domains, in the form of a ROC graph (Fawcett 2006) that plots TPR vs. FPR. As it can be seen, the performance of the goal classifier approaches the

[1] We consider two real numbers x, x' to be equal whenever $|x - x'| < \epsilon$, where ϵ is set to 10^{-7}

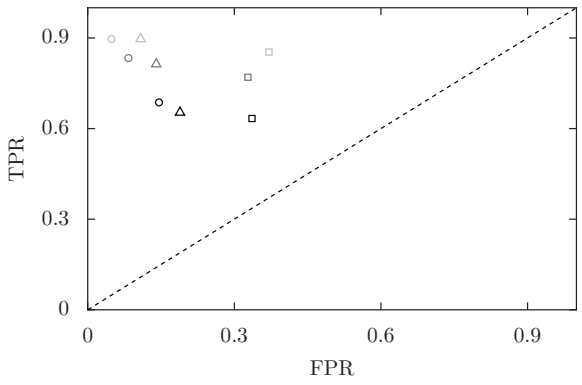

Figure 1: ROC graph showing the resulting goal classifier $\text{GR}(m, \beta)$ for different m and β values (number of samples and noise level in action selection). Squares, triangles and circles denote different m values: 100, 1000 and 10000. Black, gray, and light gray denote different β values: 1, 10, 40. Results for the random guessing strategy are represented by the dotted line.

Domain	Obs %	L	T	ACC	PPV	TPR
office	30	4.9	24.6	0.99	0.97	1.00
	50	7.6	24.7	1.00	1.00	1.00
	70	10.8	24.8	1.00	1.00	1.00
kitchen	30	3.8	95.2	0.86	0.73	0.73
	50	5.8	95.1	0.93	0.85	0.85
	70	8.3	95.2	0.98	0.95	0.95
drawers	30	2.9	38.8	0.84	0.77	0.77
	50	3.9	38.8	0.87	0.80	0.80
	70	6.0	38.8	0.96	0.93	0.93

Table 2: Performance of $\text{GR}(m = 10,000, \beta = 40)$. For each domain and observation level we report the average length of observation sequences O (L), the average time in seconds to process one observation sequence (T) and the average accuracy (ACC), precision (PPV) and True Positive rate (TPR).

optimal vertex $(1, 0)$ as the number of samples m becomes large (see caption for details). Performance is very good – high TPR, low FPR – for $m \geq 1000$ and high β values. However we can also see that even with a substantial amount of samples and the Boltzmann policy being almost greedy – $\beta = 40$ – we cannot lower FPR nor raise TPR.

Table 2 offers a detailed picture of the performance of the goal recognizer for values $m = 10,000$ and $\beta = 40$. In all domains we see how the accuracy of goal recognition increases as more information is conveyed by the input observation sequence. It is remarkable that $\text{GR}(m = 10,000, \beta = 40)$ achieves almost perfect recognition on the OFFICE domain, having some trouble with the shortest and sparsest observation sequences.

Runtime is determined by the number of possible goals $|\mathcal{G}|$, the number of samples taken m and the value for β. Processing one observation sequence involves simulating – the time required to compute $V_G(b)$ is reported on Table 1 – the Boltzmann policy m times $|\mathcal{G}|$, hence the similarity between the run-times for OFFICE and DRAWERS, which have the same number of possible goals. Runtime also increases as m increases and as β *decreases*. While the reasons for the former are quite obvious – the more simulations to check whether they are compatible with O the more computation – the former can seem a bit surprising. Runtime grows as β decreases since action selection becomes noisier, so the execution trace gets longer.

While the goal recognizing accuracy is very good, it can't be perfect since there may be observation sequences O which result ambiguous. For example, in the OFFICE domain, the goal recognizer assigns equal $P(O|G)$ to all goals, when confronted with the sequence "*walk from corridor to lab*, *walk from lab to corridor*, *walk from corridor to club*". While the joint goal is discarded because of its lower $P(G)$, there is no other information available that supports rejection of either of the singleton goals "read article" and "have coffee".

Extensions

The model above for goal recognition over POMDPs is simple but expressive, yet there are a number of natural extensions, some of which we describe next.

• *Agent POMDP model partially known by observer:* in the above formulation, the agent POMDP model is known by the observer except for the hidden goal. Incomplete information about the *initial belief state* b_0 of the agent, however, can be accommodated as well. The simplest approach is to define a set B_0 of possible initial belief states b_0 each with a probability $P(b_0)$. The formulation can then be generalized to deal with both hidden goals G and hidden initial belief states b_0, and the posterior probabilities over the collection of such pairs can be computed in a similar manner.

• *Failure to observe and actions that must be observed:* as argued in (Geib and Goldman 2009), information about actions a that if done, must always be observed, is valuable, as the absence of such actions from O, imply that their were not done. This information can be used in a direct manner in the formulation above by just adjusting the notion of when a sample execution τ complies with O. In the presence of *must-see* actions, executions τ comply with O when τ embeds O, *and* every *must-see* action appears as many times in τ as in O.

• *Observing what the agent observes:* we have assumed that the observer gets a partial trace of the actions done by the agent and nothing else. Yet, if the observer gets to see some of the observation tokens $o \in Obs$ gathered by the agent, she can use this information as well. In particular, the number m_O in (12) would then be set to the number of sampled executions for G that comply with both O and Obs.

• *Noise in the agent-observer channel:* if the observer gets to see the actions done by the agent through a noisy channel where actions can be mixed up, the problem of determining where a sample execution τ complies with the observations O is no longer a *boolean* problem where $P(O|\tau)$ is either 0 or 1, but a probabilistic inference problem that can be solved in linear-time with Hidden Markov Model (HMM) algorithms, that would yield a probability $P(O|\tau)$ in the interval $[0, 1]$. For this, the model must be extended with probabilities $P_O(o|a)$ of observing token o from the execution of

action a, and hidden chain variables $t_i = j$ expressing that the observation token o_i in $O = o_1, \ldots, o_n$ has been generated by action a_j in the sample execution $\tau = a_1, \ldots, a_m$.

Discussion

We have formulated and tested a new formulation of goal recognition for settings where the observed agent can be modeled as acting using a POMDP whose goal is hidden to the observer. The posterior goal probabilities G for the hidden goals $G \in \mathcal{G}$ are computed from Bayes rule using the priors $P(G)$ and likelihoods $P(O|G)$ that are approximated in two steps: using first a POMDP planner to produce the expected costs V_G from beliefs to goals, and using these costs to sample the possible executions for each goal G. A number of direct extensions have also been discussed, like the integration of uncertainty about the initial belief of the agent, noise in the agent-observer channel, and observations obtained by both the agent and observer. A number of experiments have been reported and the results appear promising. The software and the domains will be made available.

References

Avrahami-Zilberbrand, D., and Kaminka, G. A. 2005. Fast and complete symbolic plan recognition. In *Proceedings of IJCAI*, 653–658.

Baker, C. L.; Saxe, R.; and Tenenbaum, J. B. 2009. Action understanding as inverse planning. *Cognition* 113(3):329–349.

Barto, A.; Bradtke, S.; and Singh, S. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72:81–138.

Bertsekas, D. 1995. *Dynamic Programming and Optimal Control, Vols 1 and 2*. Athena Scientific.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proc. of AIPS-2000*, 52–61. AAAI Press.

Bonet, B., and Geffner, H. 2001. Gpt: A tool for planning with uncertainty and partial information. In *Proc. IJCAI Workshop on Planning with Uncertainty and Partial Information*.

Bonet, B., and Geffner, H. 2009. Solving POMDPs: RTDP-Bel vs. Point-based algorithms. In *Proceedings IJCAI-09*, 1641–1646.

Bui, H. 2003. A general model for online probabilistic plan recognition. In *Proc. IJCAI-03*, 1309–1318.

Cohen, P. R.; Perrault, C. R.; and Allen, J. F. 1981. Beyond question answering. In Lehnert, W., and Ringle, M., eds., *Strategies for Natural Language Processing*. LEA.

Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* (27):861–874.

Geib, C. W., and Goldman, R. P. 2009. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence* 173(11):1101–1132.

Kaelbling, L. P.; Littman, M.; and Cassandra, A. R. 1999. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101:99–134.

Kautz, H., and Allen, J. F. 1986. Generalized plan recognition. In *Proc. AAAI-86*, 32–37.

Pentney, W.; Popescu, A.; Wang, S.; Kautz, H.; and Philipose, M. 2006. Sensor-based understanding of daily life via large-scale use of common sense. In *Proc. AAAI-06*.

Pynadath, D., and Wellman, M. 2002. Generalized queries on probabilistic context-free grammars. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20(1):65–77.

Ramirez, M., and Geffner, H. 2009. Plan recognition as planning. In *Proc. 21st Intl. Joint Conf. on Artificial Intelligence*, 1778–1783. AAAI Press.

Ramirez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proc. AAAI-10*. AAAI Press.

Yang, Q. 2009. Activity Recognition: Linking low-level sensors to high-level intelligence. In *Proc. IJCAI-09*, 20–26.