In AAAI-17 Workshop on Plan, Activity and Intent Recognition (PAIR 2017)

Goal Recognition with Noisy Observations

Yolanda E-Martín[1] and **David E. Smith**[2]
[1] Centre for Automation and Robotics, CSIC-UPM, 28500 Madrid, Spain
yolanda.e.martin@csic.es
[2] Intelligent Systems Division. NASA Ames Research Center, Moffett Field, CA 94035-1000
david.smith@nasa.gov

Abstract

Goal recognition is an important technological capability for applications that involve cooperation between agents. Many goal recognition techniques allow the sequence of observations to be incomplete, but few consider the possibility of noisy observations. In this paper, we describe a planning-based goal recognition approach that deals with both missing observations and probabilistic noise in the observations. To do this, we first use a Bayesian network to infer action probabilities based on the observation model and the current observation sequence. We then use this information to estimate the expected cost of reaching the different possible goals. Comparing these costs to the a priori costs for the goals allows us to infer a probability distribution over the possible goals.

Introduction

Goal recognition is an important technological capability for applications that involve cooperation between agents. It may be that one agent needs to monitor the activities of another agent, attempt to assist the other agent, or simply avoid getting in the way while performing its own duties. For all of these cases the agent needs to be able to realize what the other agent is doing. In the absence of full and timely communication of plans and goals, goal and plan recognition becomes essential. Many goal recognition techniques allow the sequence of observations to be incomplete, but few consider the possibility of noisy observations. In practice, this is not very realistic because actions may have varying chances of being detected and correctly identified.

As an illustration of this problem, consider the ISS-CAD problem (E-Martín, R-Moreno, and Smith 2015b), where a free-flying robot is observing and documenting an astronaut performing a task in the International Space Station (ISS). Figure 1 shows a simple ISS-CAD problem, and Figure 2 a PDDL model. In particular, the problem consists of four ISS modules, Harmony (H), Destiny (D), Columbus (C), and Unity (U) between which an astronaut *a* can move to get and replace different components. The goal is to replace a sensor located in Unity.

In such a problem there might be different sources of uncertainty in the robot observations because of different reasons such as (1) occlusion: there is an obstacle between the robot and the task being performed; (2) identification: some actions may be hard to distinguish given the robot's sensors; (3) distraction: the robot is engaged in other tasks that do not allow continuous observation of the astronaut. For this reason, in this paper, we propose a planning-based goal recognition approach that deals with both missing observations and probabilistic noise in the observations. We adopt the same basic approach used in (E-Martín, R-Moreno, and Smith 2015a), which: 1) uses a plan graph to estimate costs for each possible goal given the current observation sequence, and 2) uses the Ramírez and Geffner framework (2010) to estimate the probability of each possible goal based on the difference between the cost of the best plan for the goal given the observed actions, $Cost(G|O)$, and the cost of the best plan for the goal without the observed actions, $Cost(G|\overline{O})$. The big difference here is that the observations only indirectly give us probabilities for actions in the plan graph. We therefore first construct a Bayesian Network (BN) to estimate these action probabilities, and then use this probability information in the plan graph to compute *expected* cost for each goal, given the observations.

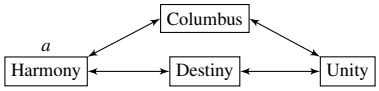


Figure 1: A simple ISS-CAD problem.

In the next section we review Ramírez and Geffner's technique for goal recognition through planning. We next describe an exhaustive but computationally impractical approach for adapting this technique to noisy observations. We then describe our novel BN cost estimation approach. Our experimental evaluation of the approach is not yet complete, but we discuss some preliminary findings. Finally, we present the conclusions and future work.

Goal Recognition Background

Ramírez and Geffner (2010) define a classical goal recognition problem as a tuple $T = \langle P, \mathcal{G}, O, Pr \rangle$ where P is a planning domain and initial conditions, $\mathcal{G}$ is a set of possible goal sets or hypotheses, O is the observed action sequence $O = o_1, ..., o_n$, and Pr is a prior probability distribution over the goal sets in $\mathcal{G}$. The observation sequence O may be incomplete, but is sequentially ordered. The solution to a goal recognition problem is the probability distribution over the goal sets $G \in \mathcal{G}$ giving the relative likelihood of

```
(define (domain ISS-CAD)
  (:requirements :strips :typing :action-costs)
  (:types crew module system component tool)
  (:predicates (connected ?m1 ?m2 - module)
               (in ?c - component ?s - system ?m - module)
               (replacement-in ?t - component ?m - module)
               (taken ?t - component ?c - crew)
               (at ?c - crew ?m - module))
  (:functions (total-cost))

  (:action move
   :parameters (?c - crew ?m1 ?m2 - module)
   :precondition (and (at ?c ?m1) (connected ?m1 ?m2))
   :effect (and (not (at ?c ?m1)) (at ?c ?m2)
                (increase (total-cost) 1))
   :observability (0.9 0.2))

  (:action get
   :parameters (?o - component ?c - crew ?m - module)
   :precondition (and (at ?c ?m) (replacement-in ?o ?m))
   :effect (and (taken ?o ?c) (increase (total-cost) 1))
   :observability (0.9 0.2))

  (:action replace
   :parameters (?o - component ?m - module ?c - crew)
   :precondition (and (at ?c ?m) (in ?o ?s ?m) (taken ?o ?c))
   :effect (and (replaced ?o ?s ?m ?c) (not (taken ?o ?c))
                (increase (total-cost) 1))
   :observability (0.9 0.2))

(define (problem ISS-CAD-p01)
  (:domain ISS-CAD)
  (:objects a - crew H D U C - module sensor - component
  (:init (connected H D) (connected D H) (connected H C)
         (connected C H) (connected D U) (connected U D)
         (connected C U) (connected U C) (taken sensor a)
         (in sensor U) (at a H) (= (total-cost) 0)
         (observable (move a H D) (1.0 0.0))
         (observable (move a D U) (0.8 0.1))
         (observable (replace sensor U a) (0.7 0.2)))
  (:goal (replaced sensor U a))
  (:metric minimize (total-cost)))
```

Figure 2: A fragment of a PDDL domain and a problem description for the ISS-CAD.

each goal set. Ramírez and Geffner estimate the probability of each possible goal set based on how compatible the observations are with plans for that goal set. In particular, they characterize the likelihood of observing O when the goal is G in terms of cost differences for achieving G under two conditions: complying with the observations O, and not complying with the observations O. Formally:

$$Pr(O|G) = \frac{e^{[-\beta\,\Delta(G,O)]}}{1 + e^{[-\beta\,\Delta(G,O)]}} \tag{1}$$

where β is a positive constant and $\Delta(G, O)$ is the cost difference between achieving the goal with and without the observations:

$$\Delta(G, O) = Cost(G|O) - Cost(G|\overline{O}) \tag{2}$$

The posterior probability $Pr(G|O)$ of a goal $G \in \mathcal{G}$ can, therefore, be calculated by computing $\Delta(G, O)$ for each possible goal. In particular, $Pr(G|O)$ can be characterized using Bayes Rule as:

$$Pr(G|O) = \alpha\, Pr(O|G)\, Pr(G) \tag{3}$$

where α is a normalization constant.

The two costs necessary to compute Δ can be found by optimally or suboptimally solving the two planning problems $G|O$ and $G|\overline{O}$.

Goal Recognition with Noisy Observations

Ramírez and Geffner's later work (2011) introduces a POMDP approach to solve goal recognition problems for an agent having partially observable actions with uncertain outcomes. This approach allows for missing observations and actions that cannot be observed, but does not allow for noise in the observations. In this work, we allow noisy observations, but we are still considering deterministic actions.

We model observation noise by the probability that the action is observed when it happens (*correct positive* CP), and the probability that the action is observed when it does not happen (*incorrect positive* IP). The syntax we use for expressing observation noise for PDDL actions is:

$$\texttt{(:observability } (CP\ \ IP)\texttt{)}$$

For the problem shown in Figure 1, observation noise for the *replace* action is modeled as:

```
(:action replace
  :parameters (?o - component ?m - module ?c - crew)
  :precondition (and (at ?c ?m) (in ?o ?m) (taken ?o ?c))
  :effect (and (replaced ?o ?m ?c) (not (taken ?o ?c))
               (increase (total-cost) 1))
  :observability (0.9 0.2))
```

However, the observability of an action often depends on the parameters. For instance, consider an astronaut opening a drawer to get a replacement component. In this case, it would be more difficult for the robot to identify that the astronaut is picking up a sensor because it is relatively small compared to a filter. For this reason, we allow specific grounded statements in the problem definition, overriding the specified observation probabilities for an action in the domain description:

```
(observable (replace sensor U a) (0.7 0.2))
```

In theory, we could use Ramírez and Geffner's framework to solve this goal recognition problem by considering all the different sequences of actions consistent with the observations, and computing a probability distribution on the goal sets for each such action sequence. Then, we could compute the weighted sum of those distributions according to the probability of the action sequence.

In the next subsections, we present the details of this approach, although it is computationally very expensive. Then, we present a much different hybrid approach that uses a Bayesian Network to infer action probabilities and a plan graph to estimate expected cost.

Exhaustive Action Sequence Approach

As before, we assume a goal recognition problem $T = \langle P, \mathcal{G}, O, Pr\rangle$, which includes a problem P, i.e., a planning domain and initial conditions with observability information, a set of possible goal sets $\mathcal{G}$, a sequence of observations O, and a prior probability distribution over $\mathcal{G}$. It is also assumed that the sequence of observed actions O may be incomplete, but is sequentially ordered.

The exhaustive solution to this problem uses optimal, or suboptimal, planners to compute an accurate likelihood of the possible goal sets. As mentioned above, for each goal set $G \in \mathcal{G}$ Ramírez and Geffner compute the difference between

$Cost(G|O)$ and $Cost(G|\overline{O})$. Strictly speaking, both costs are dependent on the noisy observations. However, in most cases, $Cost(G|\overline{O}) = Cost(G)$. The two costs only differ when the observation sequence consists exclusively of optimal landmarks (E-Martín, R-Moreno, and Smith 2015a).[1] Therefore, $Cost(G|\overline{O})$ can be approximated by solving the planning problem once for G. However, $Cost(G|O)$ is affected by the observations, so it is necessary to compute this for each possible action sequence consistent with the observations. Let $S = s_0, \ldots, s_n$ be the set of all possible action sequences consistent with the observations. In this paper, we make the assumption that only one action is possible at each time step. This assumption is not strictly necessary, but simplifies the presentation, and simplifies the computation for the exhaustive approach. To enumerate S we need to take into account all possible actions at each time step. Assume that a sequence of actions s_{t-1} at time $t-1$ has probability $P(s_{t-1})$ given the observations, and that $A_{t1}, \ldots, A_{tn}$ are the possible actions at time t. Then, the probability of an action sequence $s_t = s_{t-1}A_{tj}$ is given by:

$$P(s_t) = \begin{cases} P(s_{t-1})\text{CP}(A_{tj}) \prod_{\substack{A_{tk} \\ k \neq j}} \text{CN}(A_{tk}) & \text{if } A_{tj} \text{ is observed} \\ P(s_{t-1})\text{IN}(A_{tj}) \prod_{\substack{A_{tk} \\ k \neq j}} \text{CN}(A_{tk}) & \text{if } \emptyset \text{ is observed} \\ P(s_{t-1})\text{IN}(A_{tj})IP(A_{tk}) \prod_{\substack{A_{tm} \\ m \neq j,k}} \text{CN}(A_{tm}) & \text{if } A_{tk,k\neq j} \text{ is observed} \end{cases}$$

The set S is needed to compute $Pr(G|O)$ because it contains all the possible sequences potentially performed by the agent and observed by the robot. This means that, the problem needs to be solved for each possible action sequence $s_i \in S$. Therefore, $Pr(G|O)$ would be:

$$Pr(G|O) = \sum_{s_i \in S} Pr(G|s_i)Pr(s_i|O) \tag{4}$$

where $Pr(G|s_i)$ is computed using Equations 1, 2, and 3 with O replaced by s_i and the normalization constant α replaced by a normalization constant α_i specific to s_i, which is computed as:

$$\alpha_i = \frac{1}{\sum_{G \in \mathcal{G}} Pr(G|s_i)} \tag{5}$$

The algorithm may be summarized as: for each (possibly conjunctive) goal $G \in \mathcal{G}$:

1. Approximate $Cost(G|\overline{O}) \approx Cost(G)$ by solving the planning problem.
2. For each possible action sequence $s_i \in S$
   a. Compute $Cost(G|s_i)$ by solving the planning problem.
   b. Compute $\Delta(G, s_i)$ using Equation 2.
   c. Compute $Pr(s_i|G)$ using Equation 1.
   d. Compute $Pr(G|s_i)$ using Equation 3.
3. Compute $Pr(G|O)$ using Equation 4.

This solution provides an accurate probability distribution over the set of possible goals. However, it is computationally expensive for large problems because of the explosion in the number of action sequences consistent with the observations. To mitigate this problem, we could instead approximate the problem by using FGR (E-Martín, R-Moreno, and Smith 2015a), a goal recognition technique that avoids planning altogether by propagating cost in a plan graph. Nevertheless, both approaches have a complexity of $|\mathcal{G}| \times |S|$. For this reason, in the next section we present a much different hybrid approach that avoids enumerating the possible sequences of action S.

Hybrid Approach

Our approximate solution still uses the Ramírez and Geffner formulation (Equations 1, 2 and 3) to compute a probability distribution over the goal sets $G \in \mathcal{G}$. However, we need to find an approximate way of computing $\Delta Cost$ (Equation 2) that does not require enumerating all the action sequences consistent with the observations. If we had probability information for the actions in the plan graph, we could use this to estimate the expected cost for $Cost(G|O)$, and then use this to compute the cost difference. However, just assessing probability information for actions that are observed is not enough. The reason is that an observation may have implications about other actions and propositions. For example, if we observe an action with high probability, then there is high probability that its preconditions are true, and elevated probability for actions that could produce those preconditions at previous time steps. Conversely, the probability of mutually exclusive actions would be reduced.

In the next subsection, we describe how we automatically construct a Bayesian Network (BN) from the Plan Graph for a problem, and use this BN to infer the probabilities that actions have happened, given the current observation sequence. Then, we briefly review how to propagate cost estimates in a plan graph. Finally, we present the algorithm that uses these two techniques to compute $Cost(G|O)$ and, therefore, $Pr(G|O)$.

Estimating action probability

Plan graphs provide an efficient method of estimating the reachability of propositions and actions at different times. When actions are observed with certainty, this structure allows ruling out other actions and propositions, and inferring the presence of other actions and propositions. An example of this is the work done by Jigui and Minghao (2007), who developed a plan recognition framework that narrows the set of possible goals by incrementally pruning a plan graph as actions are observed. However, when observations are noisy, this technique is no longer sufficient. Instead, we can use a BN to infer the probability of propositions and actions in a plan graph.

A plan graph can be translated into a BN by including variables for each action and proposition, and connecting them together according to their preconditions and effects.

[1] Even when the observation sequence consists entirely of optimal landmarks, $Cost(G|\overline{O})$ and $Cost(G)$ are usually quite close. This is because the cost of the best suboptimal plan for a goal is typically close to the cost of the optimal plan.

The structure of the BN is given by the following nodes and edges:

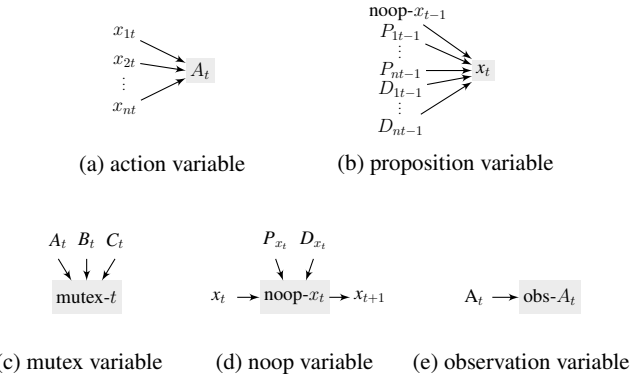
- A *proposition* variable x_0 for each proposition in the initial state. For simplicity, proposition variables that represent the initial state are given as true in the BN, (although they could be assigned uncertain values if the initial state were not fully known by the observer).
- An *action* variable A_t for each action that appears in the plan graph at time t, conditioned on all its preconditions at t. Figure 3(a) shows an action variable A_t with set of preconditions $\{x_{1t}, x_{2t}, \ldots, x_{nt}\}$.
- A *noop* variable noop-x_t for each proposition x_t, conditioned on x_t and on every other action that produces or deletes x_{t+1}. Figure 3(d) shows a noop variable for a proposition x, a set P_x of actions that can produce x (*producers*), and a set D_x of actions that can delete x (*deleters*). The noop variable has a single outgoing arc to x at the next time $t+1$.[2]
- A *proposition* variable x_t for each proposition produced at time t by actions a_{t-1} or a noop operator noop-x_{t-1}, conditioned on the producers and deleters of the proposition, and the noop operator, if it exists. Figure 3(b) shows a proposition variable x_t with producers $\{P_{1t}, \ldots, P_{nt}\}$, deleters $\{D_{1t}, \ldots, D_{nt}\}$, and the noop, noop-$x_{t-1}$.
- A *mutex* variable mutex-t at time t conditioned on all action variables a at t (noop operators not included). Mutex variables are given as true in the BN for each time step in the plan graph. Figure 3(c) shows a mutex variable where there are three action variables A_t, B_t, and C_t at time t.
- An *observation* variable obs-A_t at time t for each action A_t, conditioned on that action. Observation variables are given as true in the BN at the time step t in the plan graph where A is observed. Figure 3(e) shows an observation variable for an action variable A_t.

Figure 4 shows an example of the first few time steps of the BN for the ISS-CAD problem. Black nodes represent evidence nodes that are known as true beforehand – initial state propositions, mutex relationships, and observed actions. Gray nodes represent those variables whose state is initially unknown.

Each variable in the BN has an associated Conditional Probability Table (CPT) that gives the distribution of the variable for each combination of predecessor values. The CPT for an action variable is taken to be:

- False: if any of its preconditions is false.
- Unknown: if all its preconditions are true. In this case, we take the probability to be 1 divided by the total number of possible actions at time t. That way, all the actions whose preconditions are true at a given time step are considered equally likely, in the absence of further information.

[2]It might seen that we could just model noop variables as ordinary actions and add mutex relationships between the noop and these other actions. However, this is not enough. We also need to force the noop to be true when all other actions that affect the variable are false, which is why we need a more complex model of noop operators.



(a) action variable (b) proposition variable

(c) mutex variable (d) noop variable (e) observation variable

Figure 3: Graphical depiction of *action*, *proposition*, *mutex*, *noop*, and *observation* variables in a BN.

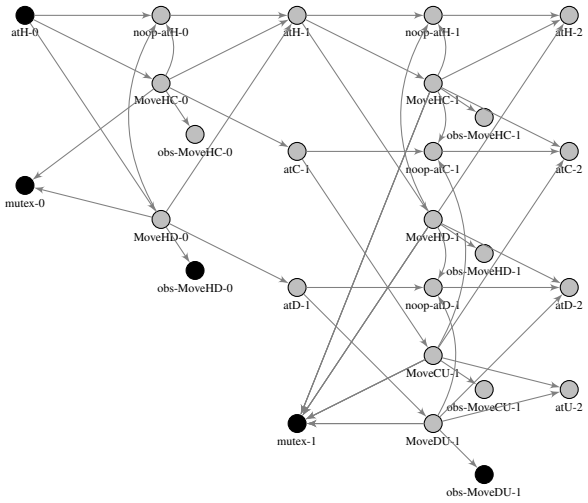


Figure 4: An fragment of the Bayesian network for the ISS-CAD problem.

Figure 5 shows the rule for the CPT of an action variable A with set of preconditions $\{x_1, x_2, \ldots, x_n\}$.

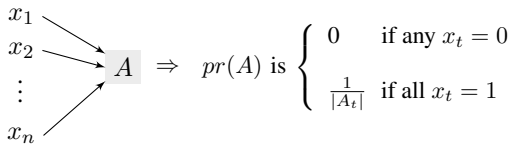


Figure 5: CPT rules for an action variable.

Table 1 illustrates the CPT of the action variable (moveHC-0) in Figure 4 whose precondition is (atH-0). When (atH-0) is true, the probability that (moveHC-0) is true or false is equal to 0.5, since there are only two actions possible at this level of the plan graph. Otherwise, the probability that (moveHC-0) is true is equal to 0, and the probability that (moveHC-0) is false is equal to 1.

The CPT for a noop variable of a proposition is taken as:

- True: if the proposition is true in the previous level and all

Table 1: CPT of an action variable (moveHC-0).

atH-0	Pr(moveHC-0=T)	Pr(moveHC-0=F)
T	0.5	0.5
F	0	1

other actions that affect the proposition are false.

- False: if the proposition is not true in the previous level or there is another action that affects the proposition.

Figure 6 shows the rule for the CPT of a noop variable noop-x for proposition x with set of producers P_x and set of deleters D_x.

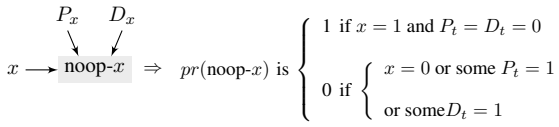


Figure 6: CPT rules for a noop variable.

Table 2 illustrates the CPT of the noop variable (noop-atH-0) in Figure 4 whose precondition is (atH-0). There are two other actions that produce (atH-0), (MoveHC-0) and (MoveHD-0), which must be considered in the CPT. In our example, when (MoveHC-0) and (MoveHD-0) are false, and (atH-0) is true, then (noop-atH-0) is true. Otherwise, (noop-atH-0) is false.

Table 2: CPT of a noop variable (noop-atH-0).

MoveHC-0	MoveHD-0	atH-0	Pr(noop-atH-0=T)	Pr(noop-atH-0=F)
T	-	-	0	1
-	T	-	0	1
F	F	T	1	0
F	F	F	0	1

The CPT for a proposition variable is taken to be:

- True: if any of the producers is true and all of the deleters are false, or all the producers and deleters are false and there is a noop.
- False: if any of the deleters is true and all of the producers are false.
- Unknown: if both a producer and deleter are true. In this case, we take the probability to be the ratio of producers to the total number of producers and deleters. That way, if a variable has many producers that are likely, but few deleters that are likely, it will be considered more likely than the opposite case.

Figure 7 shows the rule for the CPT of a proposition variable x with set of producers $\{P_1, \ldots, P_m\}$ and set of producers $\{D_1, \ldots, D_n\}$.

Table 3 illustrates the CPT of the proposition variable (atD-2) in Figure 4, whose predecessors are (MoveHD-1), (MoveDU-1), and (noop-atD-2). When (MoveHD-1) and (MoveDU-1) are true, (atD-2) is unknown and, therefore,

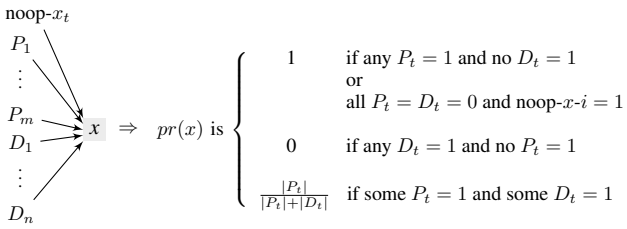


Figure 7: CPT rules for a proposition variable.

has an equal probability of 0.5 of being true or false. This is because the two actions are mutually exclusive. When (MoveHD-1) is true and (MoveDU-1) is false, (atD-2) is true with probability 1. This is because (atD-2) belongs to (MoveHD-1) add effects. Conversely, when (MoveHD-1) is false and (MoveDU-1) is true, (atD-2) is true with probability 0. This is because (atD-2) belongs to (MoveDU-1) delete effects. Finally, when (MoveHD-1) and (MoveDU-1) are false and (noop-atD-1) is true, (atD-2) is true with probability 1.

Table 3: CPT of the proposition variable (atD-2).

MoveHD-1	MoveDU-1	noop-atD-1	Pr(atD-2=T)	Pr(atD-2=F)
T	T	-	0.5	0.5
T	F	-	1	0
F	T	-	0	1
F	F	T	1	0
F	F	F	0	1

The CPT for a mutex variable among n actions $A_{0i} \ldots A_{nt}$ at time t is defined as:

- True: if at most one A_{it} is true.
- False: if more than one A_{it} is true.

Figure 8 shows the rule for the CPT of a mutex variable mutex-t for actions A_{t1}, A_{t2}, and A_{t3}.

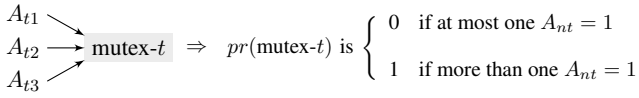


Figure 8: CPT rules for a mutex variable.

By setting all mutex variables to true in the BN, we prevent more than one action at any given time step. This is not a limitation for the hybrid approach, but it is for the exhaustive action sequence approach that we want to compare against.

Table 4 illustrates the CPT of the mutex variable (mutex-0) in Figure 4.

Finally, the CPT of an observation variable is generated using the probability information given by the *observable* statement in the problem definition. Figure 9 shows the rule for the CPT of an observable variable obs-A for an action A.

Table 5 shows the CPT of the observation variable (obs-MoveDU-1) in Figure 4, which is observed at time *1* with a

Table 4: CPT of a mutex variable (mutex-0).

MoveHC-0	MoveHD-0	Pr(mutex-0=T)	Pr(mutex-0=F)
T	T	0	1
-	F	1	0
F	-	1	0

$$A \longrightarrow \text{obs-}A \;\Rightarrow\; pr(\text{obs-}A) \text{ is } \begin{cases} \textit{Correct positive} & \text{if } A = 1 \\ \textit{Incorrect positive} & \text{if } A = 0 \end{cases}$$

Figure 9: CPT rules for an observation variable.

probability of 0.8 of being observed if true, and a probability of 0.1 of being observed if false.

Table 5: CPT of an observation variable (obs-MoveDU-1).

MoveDU-1	Pr(obs-MoveDU-1=T)	Pr(obs-MoveDU-1=F)
T	0.8	0.2
F	0.1	0.9

Once we generate the BN for the problem, we can query the posterior probability of each action and proposition given all evidence variables – that is, the actions in the observed sequence. In our example, we know by intuition that the probability of (atD-1) should be higher than the probability of (atH-1) and (atC-1); and that the probability of (atU-2) should be higher than the probability of (atD-2) because actions (MoveHD-0) and (MoveDU-1) are observed with probabilities 1 and 0.8 respectively. In particular, when we query variables (atH-1), (atC-1), and (atD-1), we get the following non-zero posterior probability:

$$\begin{aligned} &\text{Pr(atD-1=T} \,|\, \text{evidence)} = 1 \\ &\text{Pr(atD-2=T} \,|\, \text{evidence)} = 0.015 \\ &\text{Pr(atU-2=T} \,|\, \text{evidence)} = 0.985 \end{aligned}$$

If, for instance, we considered the case where the robot only observed action (MoveHD-0), propositions (atD-2) and (atU-2) would have a 0.5 posterior probability of being true. This is because there is not enough information or evidence to infer the astronaut behavior, and, therefore, both locations would have the same probability.

Cost estimation in a plan graph

The standard method of propagating cost information in a plan graph is a technique that has been used in a number of planning systems (e.g: (Do and Kambhampati 2002)), and assumes independence among all preconditions of an action. The computation of cost begins at level zero of a plan graph and proceeds sequentially to higher levels. For level zero we assume the cost for propositions is 0 because the initial state is given. With these assumption, the propagation starts by computing the cost of the actions at level zero. In general, for an action a at level l with a set of preconditions $\mathcal{P}_a$, the cost is approximated as:

$$cost(a) = cost(\mathcal{P}_a) = \sum_{x_t \in \mathcal{P}_a} cost(x_t) \tag{6}$$

The next step is to calculate the cost of propositions. For a proposition x at level l, achieved by the actions $\mathcal{A}_x$ at the preceding level, the cost is calculated as:

$$Cost(x) = \min_{a \in \mathcal{A}(x)} [cost(a) + Cost_a] \tag{7}$$

where $Cost_a$ is the cost of applying the action a, and $cost(a)$ is the cost of achieving a, given by Equation 6.

Using these equations, a cost-plan graph is built until quiescence. On completion, each possible goal proposition has an estimated cost of achievement. For each possible conjunctive goal $G \in \mathcal{G}$, a relaxed plan is computed, giving preference at each step of the regression to the producer actions (or Noops) having the lowest cost. The cost of this relaxed plan is then taken as $Cost(G)$.

The nFGR algorithm

We can use the posterior probability values computed in the BN to help infer the expected cost of achieving various goals. To do this, we take into account the BN probabilities of the actions in the propagation of cost in a plan graph. The plan graph can, therefore, be used to approximate an expected cost for each goal. In particular, we follow the previously described algorithm for propagating cost in a plan graph. That is, the cost of an action is the cost of achieving its preconditions. The cost of propositions is the minimum cost among all the actions that achieve the proposition. However, it is possible that a costly action has a high probability of being true given the observed action sequence. As a consequence, the probability of propositions depend on the probability of actions being true. For this reason, we make use of the posterior probabilities pr given by the BN to infer the expected cost of propositions and actions in a plan graph. As before, for level zero we assume the cost for propositions at this level is 0 because the initial state is given. The propagation starts by computing the cost of the actions at level zero.

In general, the cost of an action is the sum of the costs of achieving its preconditions. However, in the probabilistic case, the cost of an action will only be the cost of those preconditions that are not already established. We can, therefore, use the probabilities computed in the BN to temper the costs of preconditions, based on how likely it is that they have already been achieved. In this way, for a positive precondition p, the cost is multiplied by $1 - pr(p)$, while for a negative preconditions q, it is multiplied by $pr(q)$. In general, for an action a at level l with a set of positive preconditions $\mathcal{P}_a$, and a set of negative preconditions $\mathcal{N}_a$, the expected cost of a is approximated as:

$$eCost(a) = \sum_{p \in \mathcal{P}_a} eCost(p)\{1 - pr(p)\} + \sum_{p \in \mathcal{N}_a} eCost(p)pr(p) \tag{8}$$

The next step is to calculate the cost of propositions. For a proposition x at level l, achieved by the actions $\mathcal{A}_x$ at the preceding level, the expected cost of x is approximated as:

$$eCost(x) = cost(x) \prod_{a \in \mathcal{A}_x} (1 - pr(a)) \tag{9}$$

where $cost(x)$ is given by Equation 6, and the product of all producers of x is the probability that none of the producers has been performed. In other words, we do not have to pay any cost for those actions that have been performed. It is only those actions that are not done yet that influence the cost of achieving x, and it is the cost times $1 - pr(a)$.

Taking these decision rules into consideration, we can build a plan graph and propagate expected cost. The construction process finishes when two consecutive proposition layers are identical and there is quiescence in cost for all propositions and actions in the plan graph. On completion, each possible goal proposition has an expected cost of being achieved.

For each possible conjunctive goal $G \in \mathcal{G}$, we then construct a relaxed plan π, giving preference at each step of the regression to the producer actions (or Noops) having the lowest expected cost. The expected cost of $G|O$ is then taken to be the cost of this relaxed plan π plus the expected cost of other actions in the plan graph that are not present in the relaxed plan π but have non-zero probability. The high-level nFGR algorithm used to solve a goal recognition problem for a particular sequence of observations is summarized in the following steps:

1. Build a plan graph for the problem P (domain plus initial conditions) and propagate cost through this plan graph.
2. For each (possibly conjunctive) goal $G \in \mathcal{G}$ estimate the $Cost(G)$ from a relaxed plan.
3. Build a BN, using the technique described above.
4. For the particular observation sequence $o_1, ..., o_t$ set the corresponding observation variables in the BN and use it to estimate posterior probability for actions and propositions.
5. Using the probabilities from the BN, compute expected costs for all possible goals using the probabilistic plan graph.
6. For each (possibly conjunctive) goal $G \in \mathcal{G}$:
   a. Estimate the $Cost(G|O)$ from a relaxed plan.
   b. Compute $\Delta(G, O)$ using Equation 2, and compute the probability $Pr(G|O)$ for the goal given the observations using Equation 1.
7. For each (possibly conjunctive) goal $G \in \mathcal{G}$ compute $Pr(O|G)$ as in Equation 3.

When used for real-time goal recognition, the plan graph and the BN are built once. Steps 4 to 7 are repeated every time an action is observed. In other words, every time an observation comes in, the corresponding observation variable is set to true in the BN, and the posterior probability for each action and proposition is recomputed. Then, the cost propagation is done again in the plan graph to compute the expected cost of each possible goal.

Discussion

We have developed an implementation of the Exhaustive Action Sequences approach described earlier in this paper. Unfortunately this approach requires large amounts of computation time to get results, due to the combinatorial explosion in the number of possible action sequences that need to be considered. We attempted to run this approach using both the optimal planner HSP$^*_f$ (Haslum and Geffner 2000) and our heuristic approximation FGR (E-Martín, R-Moreno, and Smith 2015a). HSP$^*_f$ runs out of time for most of the problems with a time limit of 3600 seconds. FGR runs out of memory or runs out of time with a time limit of 1800 seconds. Clearly these times are much too high to allow use of this technique for real-time goal recognition. However, we would nevertheless like to obtain results with this approach (however long it takes) to compare the accuracy of the predications with our hybrid nFGR technique.

For the hybrid approach, nFGR, our initial implementation used an open source Java BN package. This package turned out to have efficiency problems for the large networks we create. In addition, mutex relationships and noop operations tend to have large but very sparse CP tables. The BN package performs poorly with these large tables. As a result, we are currently adapting our implementation to use a more powerful commercial BN package that allows us to specify sparse CP tables using equations. We believe that this package will perform much better, but have not yet completed implementation and testing.

Related work

As previously mentioned, Ramírez and Geffner (2011) introduce a POMDP approach to solve goal recognition problems for an agent having partially observable actions with uncertain outcomes. This approach allows for missing observations and actions that cannot be observed, but does not allow for noise in the observations. More recent work by Keren, Gal, and Karpas (2016) uses a model of sensing where performance of an action nondeterministically results in one of a set of possible sensory outcomes (tokens) that may or may not be unique to the action. This is somewhat different from the model we use here, where we assume true-positive and false-positive probabilities of observation for each action. Given a probability distribution over the sensor tokens for each action, it appears possible to compile the sensor token model into our model, but it is less clear whether it is possible to go the other direction.

Bayesian networks have become a popular representation to perform inference based on observed actions and a knowledge-based model. In particular, Charniak and Goldman (1993) developed a framework to solve plan recognition problems, which consists of a knowledge-base of facts about the world expressed in a first-order language and rules for using that knowledge-base to build a BN. The network is then evaluated to find the plans consistent with the observed sequence with the highest probability. Albrecht and colleagues (1997) implemented online goal recognition for adventure games using Dynamic Bayesian Networks (DBN) (which allows incorporating temporal reasoning into the BN) to recognize possible goals and to predict future player actions. Horvitz and Paek (1999) developed an approach that uses BN to recognize goals in an automated conversation

system. Kaminka and colleagues (2002) developed an approach to multiagent plan recognition using DBN to perform monitoring in distributed systems. Bui (2003) and Bui and colleagues (2002) used Hierarchical Hidden-Markov Models, a specific type of DBN, for hierarchical goal recognition, which is the recognition of the current agents top-level goal and subgoals. Saria and Mahadevan (2004) extended the work by Bui (2003) to multiagent plan recognition. Charniak and Goldman (2009) developed a Bayesian approach to plan recognition that evaluates abductive story understanding systems.

For most of these plan recognition and goal recognition approaches, the BN is built in a domain specific fashion. The one exception is the work done by Charniak and Goldman (1993). They provide a set of rules, which define the preconditions for adding nodes to the network, with information on actions, plans, and observed objects. The network grows with every new observation and by creating new hypotheses. The result is a posterior probability distribution over a set of possible plans. This technique is quite different from the approach we use, where the BN is constructed directly from the planning domain model. In addition to this, we do not use the BN to directly infer the probability of the possible goals. Instead, we use the BN to infer action and proposition probabilities, which are then used to estimate expected cost for each possible goal using a plan graph. This cost information is then used with the Ramírez and Geffner approach to infer a probability distribution over the possible goals.

Conclusions and future work

This paper presented a heuristic technique for goal recognition with noisy observations. It combines the construction of a plan graph with cost estimates (E-Martín, R-Moreno, and Smith 2015a) and Bayesian networks (Pearl 1988) to compute likelihood of the possible goals. In particular, we use the posterior probability values computed in the BN to help infer the expected cost of achieving various goals. Our experimental evaluations are still ongoing, so it is early to draw significant conclusions about the accuracy and efficiency of this work. However, preliminary results show that the Exhaustive Action Sequences approach is impractical for real-time goal recognition purposes.

In addition to finishing our experimental evaluation, there is another issue that we would like to explore. The simple cost propagation in a plan graph is a technique that assumes independence between propositions and between actions. Therefore, the cost estimates computed in this way are often inaccurate because they do not consider interaction between different actions and subgoals. In our earlier work on FGR (E-Martín, R-Moreno, and Smith 2015a), we addressed this issue for goal recognition with certain observations. However, noisy observations lead to probabilities on actions and propositions in the plan graph, which complicates the computation of interaction estimates. As a result, we have ignored interaction computation in this work. A natural extension is to consider interaction during cost propagation along with the probabilistic information given by the BN. In this way, we could compute more accurate estimates of the cost of a goal, and the expected cost of a goal given the observations.

Acknowledgments

We thank Mark Peot for discussions on the encoding of a plan graph in a Bayesian Network and Christian Plaunt for his help running preliminary experiments. This work is partially supported by the NASA Safe Autonomous Systems Operations (SASO) project.

References

Albrecht, D. W.; Zukerman, I.; Nicholson, A. E.; and Bud, A. 1997. Towards a Bayesian model for keyhole plan recognition in large domains. In *6th Intl. Conf. on User Modeling*.

Bui, H.; Venkatesh, S.; and West, G. 2002. Policy recognition in abstract hidden Markov model. *JAIR* 17(4):51–99.

Bui, H. 2003. A general model for on-line probabilistic plan recognition. In *IJCAI-03*.

Charniak, E., and Goldman, R. P. 1993. A Bayesian model of plan recognition. *AIJ* 64(1):53–79.

Do, M., and Kambhampati, S. 2002. Planning graph-based heuristics for cost-sensitive temporal planning. In *AIPS-02*.

E-Martín, Y.; R-Moreno, M. D.; and Smith, D. E. 2015a. A fast goal recognition technique based on Interaction estimates. In *IJCAI-15*.

E-Martín, Y.; R-Moreno, M. D.; and Smith, D. E. 2015b. Practical goal recognition for ISS Crew Activities. In *IWPSS-15*.

Geib, C. W., and Goldman, R. P. 2009. A probabilistic plan recognition algorithm based on plan tree grammar. *AIJ* 84(1-2):57–112.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *AIPS-00*.

Horvitz, E., and Paek, T. 1999. A computational architecture for conversation. *7th International Conference on User Modeling*.

Jigui, S., and Minghao, Y. 2007. Recognizing the agent's goals incrementally: planning graph as a basis. *Frontiers of Computer Science in China* 1(1):26–36.

Kaminka, G. A.; Pynadath, D. V.; and Tambe, M. 2002. Monitoring teams by overhearing: A multi-agent plan recognition approach. *JAIR* 17:83–135.

Keren, S.; Gal, A.; and Karpas, E. 2016. Privacy preserving plans in partially observable environments. In *IJCAI-16*.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

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

Ramírez, M., and Geffner, H. 2011. Goal recognition over POMDPs. In *ICAPS-11*.

Saria, S., and Mahadevan, S. 2004. Probabilistic plan recognition in multiagent systems. In *ICAPS-04*.