Plan Recognition as Planning

Miquel Ramírez
Universitat Pompeu Fabra
08003 Barcelona, SPAIN
miquel.ramirez@upf.edu

Hector Geffner
ICREA & Universitat Pompeu Fabra
08003 Barcelona, SPAIN
hector.geffner@upf.edu

Abstract

In this work we aim to narrow the gap between plan recognition and planning by exploiting the power and generality of recent planning algorithms for recognizing the set $\mathcal{G}^*$ of goals G that explain a sequence of observations given a domain theory. After providing a crisp definition of this set, we show by means of a suitable problem transformation that a goal G belongs to $\mathcal{G}^*$ if there is an action sequence π that is an optimal plan for both the goal G and the goal G extended with extra goals representing the observations. Exploiting this result, we show how the set $\mathcal{G}^*$ can be computed exactly and approximately by minor modifications of existing optimal and suboptimal planning algorithms, and existing polynomial heuristics. Experiments over several domains show that the suboptimal planning algorithms and the polynomial heuristics provide good approximations of the optimal goal set $\mathcal{G}^*$ while scaling up as well as state-of-the-art planning algorithms and heuristics.

1 Introduction

Plan recognition is an ubiquitous task in a number of areas, including natural language, multi-agent systems, and assisted cognition [Schmidt *et al.*, 1978; Cohen *et al.*, 1981; Pentney *et al.*, 2006]. Plan recognition is planning in reverse: while in planning, we seek the actions to achieve a goal, in plan recognition, we seek the goals that explain the observed actions. Both are abductive tasks where some hypothesis needs to be adopted to account for given data: plans to account for goals, and goals to account for partially observed plans. Work in plan recognition, however, has proceeded independently of the work in planning, using mostly handcrafted libraries or algorithms that are not related to planning [Kautz and Allen, 1986; Vilain, 1990; Charniak and Goldman, 1993; Lesh and Etzioni, 1995; Goldman *et al.*, 1999; Avrahami-Zilberbrand and Kaminka, 2005].

In this work we aim to narrow the gap between plan recognition and planning by exploiting the power and generality of recent classical planning algorithms. For this, we move away from the plan recognition problem over a *plan library* and consider the plan recognition problem over a *domain theory* and a possible set $\mathcal{G}$ of goals. The plan recognition task is then formulated as the problem of identifying the goals $G \in \mathcal{G}$ such that some *optimal* plan for G is compatible with the observations. Such goals are grouped into the set $\mathcal{G}^*$. The reason for focusing on the optimal plans for G is that they represent the possible behaviors of a perfectly rational agent pursuing the goal G. By a suitable transformation, we show that a goal G is in $\mathcal{G}^*$ if there is an action sequence that achieves optimally *both* the goal G and the goal G extended with extra goals G_o representing the observations. Exploiting this result, we show how the set of goals $\mathcal{G}^*$ can be computed *exactly* and *approximately* by minor modifications of existing optimal and suboptimal planning algorithms, and existing polynomial heuristics, that are then tested experimentally over several domains. The suboptimal algorithms and the heuristics appear to provide good approximations of the optimal goal set $\mathcal{G}^*$, while scaling up as well as state-of-the-art planning algorithms and heuristics.

The paper is organized as follows. We provide first the basic planning terminology, define the plan recognition problem over a domain theory, and introduce the transformation that allows us to compile away the observations. We then focus on exact and approximate computational methods, present the result of the experiments, and discuss limitations and generalizations of the proposed framework.

2 Planning Background

A Strips planning problem is a tuple $P = \langle F, I, A, G \rangle$ where F is the set of fluents, $I \subseteq F$ and $G \subseteq F$ are the initial and goal situations, and A is a set of actions a with precondition, add, and delete lists $Pre(a)$, $Add(a)$, and $Del(a)$ respectively, all of which are subsets of F.

For each action $a \in A$, we assume that there is a non-negative cost $c(a)$ so that the cost of a sequential plan $\pi = a_1, \ldots, a_n$ is $c(\pi) = \sum c(a_i)$. A plan π is *optimal* if it has minimum cost. For unit costs, i.e., $c(a) = 1$ for all $a \in A$, plan cost is plan length, and the optimal plans are the shortest ones. Unless stated otherwise, action costs are assumed to be 1.

In the plan recognition setting over a domain theory, action costs $c(a)$ carry implicit information about the probability that the agent will use action a to solve a problem P, as the agent should rationally avoid plans with higher cost than

needed. The most likely plans, assuming full rationality, are the optimal plans, and hence, we will focus initially on them.

Algorithms for computing optimal plans cast the planning problem as a problem of heuristic search with *admissible* heuristics derived automatically from the problem encoding [Haslum and Geffner, 2000] . An heuristic $h(s)$ is admissible if $h(s) \leq h^*(s)$ for all the states, where $h^*(s)$ is the optimal cost from s. Optimal planners use such heuristics in the context of admissible search algorithms like A* and IDA*. Computing optimal plans, however, is much harder than computing plans that are good but not necessarily optimal, and indeed suboptimal planners such as FF and FD, that use more focused search algorithms and more informative heuristics, scale up much better [Hoffmann and Nebel, 2001; Helmert, 2006]. We will thus use optimal plans for providing a *crisp definition* of the plan recognition task, and develop *efficient methods* that borrow from suboptimal planning methods and heuristics for producing good approximations that scale up.

3 Plan Recognition over a Domain Theory

The plan recognition problem given a *plan library* for a set $\mathcal{G}$ of possible goals G can be understood, at an abstract level, as the problem of finding a goal G with a plan π in the library, written $\pi \in \Pi_L(G)$, such that π satisfies the observations. We define the plan recognition problem over a *domain theory* in a similar way just changing the set $\Pi_L(G)$ of plans for G in the library by the set $\Pi^*_P(G)$ of optimal plans for G given the domain P. We will use $P = \langle F, I, O\rangle$ to represent planning *domains* so that a planning *problem* $P[G]$ is obtained by concatenating a planning domain with a goal G, which is any set of fluents. We can then define a *plan recognition problem* or *theory* as follows:

Definition 1. *A plan recognition problem or* theory *is a triplet $T = \langle P, \mathcal{G}, O\rangle$ where $P = \langle F, I, A\rangle$ is a planning domain, $\mathcal{G}$ is the set of possible goals G, $G \subseteq F$, and $O = o_1, \ldots, o_m$ is an* observation sequence *with each o_i being an action in A.*

We also need to make precise what it means for an action sequence to satisfy an observation sequence made up of actions. E.g., the action sequence $\pi = \{a, b, c, d, e, a\}$ satisfies the observation sequences $O_1 = \{b, d, a\}$ and $O_2 = \{a, c, a\}$, but not $O_3 = \{b, d, c\}$. This can be formalized with the help of a function that maps observation indices in O into action indices in A:

Definition 2. *An action sequence $\pi = a_1, \ldots, a_n$ satisfies the observation sequence $O = o_1, \ldots, o_m$ if there is a* monotonic *function f mapping the observation indices $j = 1, \ldots, m$ into action indices $i = 1, \ldots, n$, such that $a_{f(j)} = o_j$.*

For example, the unique function f that establishes a correspondence between the actions o_i observed in O_1 and the actions a_j in π is $f(1) = 2$, $f(2) = 4$, and $f(3) = 6$. This function must be strictly monotonic so that the action sequences π preserve the ordering of the actions observed.

The solution to a plan recognition theory $T = \langle P, \mathcal{G}, O\rangle$ is given then by the goals G that admit an optimal plan that is compatible with the observations:

Definition 3. *The exact solution to a theory $T = \langle P, \mathcal{G}, O\rangle$ is given by the* optimal goal set $\mathcal{G}^*_T$ *which comprises the goals $G \in \mathcal{G}$ such that for some $\pi \in \Pi^*_P(G)$, π satisfies O.*

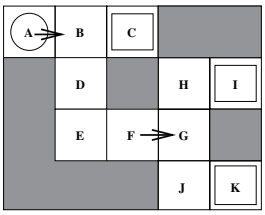

Figure 1: Plan Recognition in a Navigation Domain

Figure 1 shows a simple plan recognition problem. Room A (marked with a circle) is the initial position of the agent, while Rooms C, I and K (marked with a square) are its possible destinations. Arrows between Rooms A and B, and F and G, are the observed agent movements in that order. In the resulting theory T, the only possible goals that have optimal plans compatible with the observation sequence are I and K. In the terminology above, the set of possible goals $\mathcal{G}$ is given by the atoms $at(C)$, $at(I)$, and $at(K)$, while the optimal goal set $\mathcal{G}^*_T$ comprises $at(I)$ and $at(K)$, leaving out the possible goal $at(C)$.

Before proceeding with the methods for computing the optimal goal set $\mathcal{G}^*_T$ exactly or approximately, let us first comment on some of the limitations and strengths of this model of plan recognition. The model can be easily extended to handle observations on fluents and not only on actions. For this, the observation of a fluent p can be encoded as the observation of a 'dummy' action $NOOP(p)$ with precondition and effect p. Similarly, it is not difficult to account for actions that *must* be among the observations when they have been executed. For this, it suffices to remove such actions from the domain theory when they haven't been observed. Notice that the same tricks do not work in library-based approaches that are less flexible. Likewise, the assumption that the observations are ordered in a linear sequence is not critical, and can be replaced with no much trouble by a partial order. Library-based recognition can also be accommodated by a suitable compilation of (acyclic) libraries into Strips [Lekavý and Návrat, 2007]. A critical aspects of the model is that it does not weight the hypotheses G in $\mathcal{G}$ but it just filters them. We will talk more about this in Section 8.

4 Compiling the Observations Away

In order to solve the plan recognition problem using planning algorithms, we need to get rid of the observations. This is actually not difficult to do. For simplicity, we will assume that no pair of observations o_i and o_j refer to the same action a in P. When this is not so, we can create a copy a' of the action a in P so that o_i refers to a' and o_j refers to a.

We will compile the observations away by mapping the theory $T = \langle P, \mathcal{G}, O\rangle$ into an slightly different theory $T' = \langle P', \mathcal{G}', O'\rangle$ with an empty set O' of observations, such that the solution set $\mathcal{G}^*_T$ for T can be read off from the solution set $\mathcal{G}^*_{T'}$ for T'.

Definition 4. *For a theory $T = \langle P, \mathcal{G}, O\rangle$, the transformed theory is $T' = \langle P', \mathcal{G}', O'\rangle$ such that*

- *$P' = \langle F', I', A'\rangle$ has fluents $F' = F \cup F_o$, initial situation $I' = I$, and actions $A' = A \cup A_o$, where $P = \langle F, I, A\rangle$, $F_0 = \{p_a \mid a \in O\}$, and $A_o = \{o_a \mid a \in O\}$,*
- *$\mathcal{G}'$ contains the goal $G' = G \cup G_o$ for each goal G in $\mathcal{G}$, where $G_o = F_o$,*
- *O' is empty*

The new actions o_a in P' have the same precondition, add, and delete lists as the action a in P except for the new fluent p_a that is added to $Add(o_a)$, and the fluent p_b, for the action b that immediately precedes a in O, if any, that is added to $Pre(o_a)$.

In the transformed theory T', the observations $a \in O$ are encoded as extra fluents $p_a \in F_o$, extra actions $o_a \in A_o$, and extra goals $p_a \in G_o$. Moreover, these extra goals p_a can only be achieved by the new actions o_a, that due to the precondition p_b for the action b that precedes a in O, can be applied only after all the actions preceding a in O, have been executed. The result is that the plans that achieve the goal $G' = G \cup G_o$ in P' are in correspondence with the plans that achieve the goal G in P *that satisfy the observations* O:

Proposition 5. *$\pi = a_1, \ldots, a_n$ is a plan for G in P that satisfies the observations $O = o_1, \ldots, o_m$ under the function f iff $\pi' = b_1, \ldots, b_n$ is a plan for G' in P' with $b_i = o_{a_i}$, if $i = f(j)$ for some $j \in [1, m]$, and $b_i = a_i$ otherwise.*

It follows from this that π is an optimal plan for G in P that satisfies the observations iff π' is an optimal plan in P' for two different goals: G, on the one hand, and $G' = G \cup G_o$ on the other. If we let $\Pi^*_P(G)$ stand for the set of optimal plans for G in P, we can thus test whether a goal G in $\mathcal{G}$ accounts for the observation as follows:[1]

Theorem 6. *$G \in \mathcal{G}^*_T$ iff there is an action sequence π in $\Pi^*_{P'}(G) \cap \Pi^*_{P'}(G')$.*

Moreover, since $G \subseteq G'$, if we let $c^*_{P'}(G)$ stand for the optimal cost of achieving G in P', we can state this result in a simpler form:

Theorem 7. *$G \in \mathcal{G}^*_T$ iff $c^*_{P'}(G) = c^*_{P'}(G')$*

As an example of the transformation, for the plan recognition task shown in Figure 1, the extra fluents F_o in T' are $p_{move(A,B)}$ and $p_{move(F,G)}$, while the extra actions A_o are $o_{move(A,B)}$ and $o_{move(F,G)}$; the first with the same precondition as $move(A, B)$ but with the extra fluent $p_{move(A,B)}$ in the Add list, and the second with the extra precondition $p_{move(A,B)}$ and the extra effect $p_{move(F,G)}$. Theorem 7 then means that a goal G accounts for the observations in the original theory T, and thus belongs to $\mathcal{G}^*_T$, iff the cost of achieving the goal G in the transformed domain P' is equal to the cost of achieving the goal $G' = G \cup G_o$. *In the transformed problem, thus, observations have been replaced by extra goals that must be achieved at no extra cost.*

[1] Note that while a plan for $G' = G \cup G_o$ is always a plan for G, it is *not* true that an *optimal* plan for G' is an *optimal* plan for G, or even that a good plan for G' is a good plan for G.

5 Computation

We present next a method for computing the optimal goal set $\mathcal{G}^*_T$ exactly, and two methods that approximate this set but scale up better. The first method is based on domain-independent optimal planning techniques, while the latter two are based on suboptimal planning algorithms and heuristic estimators respectively.

5.1 Exact Methods

Optimal planning algorithms are designed for computing an action sequence in $\Pi^*_P(G)$, if one exists, for any planning domain P and goal G. The cost of such a plan is the optimal cost $c^*_P(G)$. This computation is done by running an admissible search algorithm like A* or IDA* along with an admissible heuristic h that is extracted automatically from the problem [Haslum and Geffner, 2000].

Theorem 7 can be used to check whether a goal G is in $G \in \mathcal{G}^*_T$ using an optimal planner for computing the optimal cost $c^*_{P'}(G)$ of achieving G in P', and then using this cost as an *upper bound* in the search for an optimal plan for the goal $G' = G \cup G_o$ in P'. This computation is not exactly the solution to two planning problems as the search in the second problem is heavily pruned by the use of cost $c^*_{P'}(G)$ as a bound. The bound can be used profitably in either A* or IDA* searches, in both cases, pruning right away nodes n with cost $f(n) = g(n) + h(n)$ greater than $c_{P'}(G)^*$. In the experiments below, we use the latest implementation of the optimal HSP* planner due to Haslum, that uses the A* algorithm modified to accommodate this pruning.

Three other optimizations have been made in the code for HSP*. First, the cost $c^*_{P'}(G)$ of achieving the goal G in P' is equivalent to the cost $c^*_P(G)$ of achieving the same goal in the original problem P that has no extra fluents or actions, and thus, can be solved more efficiently. This is because every occurrence of an actions o_a in P' that is not in P can be replaced by the corresponding action a in P. Second, since the difference between the action o_a in P' and the action a in P is that o_a adds the fluent p_a that is never deleted, it is not necessary to consider plans for G' in P' where such actions are done more than once. And last, since there cannot be plans for G' in P' with cost lower than $c^*_{P'}(G)$, in the second part of the computation we are just testing whether there is one such plan for G' with cost equal to $c^*_{P'}(G)$. For this, it is wasteful to use A*; it is more efficient to run a single IDA* iteration with memory and setting the initial bound to $c^*_{P'}(G)$, avoiding the use of a heap.

5.2 Approximate Methods

Suboptimal planners such as FF and FD scale up to much larger problems than optimal planners as they can use more informed heuristics that are not necessarily admissible and more focused search algorithms. We want to use such suboptimal planners for computing approximations of the optimal goal set $\mathcal{G}^*_T$. This approximation can be understood as replacing the test that checks the existence of an action sequence π in $\Pi^*_{P'}(G) \cap \Pi^*_{P'}(G')$, by the weaker test that checks for the presence of an action sequence π in $\Pi^s_{P'}(G) \cap \Pi^s_{P'}(G')$, where $\Pi^s_P(G)$ denotes a set of 'good plans' for G not neces-

sarily optimal. For example, while planners such as FF produce a *single plan* for a goal G in a domain P, we can think of $\Pi^s_P(G)$ as the *set* of plans that FF is capable of producing if ties were broken in all possible ways. Yet, we do not need to make this intuition precise because we will not compute such sets of plans explicitly: rather we will modify FF so that it will 'break ties' for producing a *single* plan π_G for G in P' that achieves as many goals from G' (observations) as (greedily) possible.

The FF planner in Enforced Hill Climbing (EHC) mode iteratively moves from one state s to another state s' with lower heuristic value h_{FF} that is found by a local breadth first search from s that considers a small fragment of the actions applicable in a state, called the helpful actions [Hoffmann and Nebel, 2001]. Both the helpful actions in s and the heuristic value $h_{\text{FF}}(s)$ are derived from the relaxed plan $\pi(s)$ that is obtained efficiently from every state s explored.

In this process, there are *two decision points* where the extra goals $p_a \in G' \setminus G$, encoding the observations in O, can be used to bias the search towards plans that achieves as many atoms p_a from G' as possible. First, rather than moving from s to the first state s' that improves the heuristic value $h_{\text{FF}}(s)$ of s, we exhaust the local breadth first search space bounded by $g(s')$, and commit to the state s'' in this local space that improves the heuristic value of s and maximizes the measure $|\pi(s'') \cap A_o|$. Second, rather than computing the relaxed plans $\pi(s)$ as in FF, we follow the formulation of the set-additive heuristic in [Keyder and Geffner, 2008], where the relaxed plans $\pi(p; s)$ for all fluents p from the state s are computed recursively. For biasing the construction of the relaxed plan $\pi(s)$ in s for accounting for as many dummy goals p_a in G' as possible, we just choose as the 'best supporter' a_p of a fluent p, the action a that adds p and minimizes $c(\pi(a; s))$, while maximizing, in addition, the measure $|\pi(a; s) \cap A_o|$, where $\pi(a; s) = \{a\} \cup \cup_{q \in Pre(a)} \pi(q; s)$.

With this modified version of the FF planner, we define the approximation $\mathcal{G}^s_T$ of the optimal goal set $\mathcal{G}^*_T$, as the set of goals G in $\mathcal{G}$ for which the measure $|\pi_{P'}(G) \cap A_o|$ is highest, where $\pi_{P'}(G)$ is the plan produced by the modified FF planner for achieving the goal G in P'.

We will actually test below a second approximation $\mathcal{G}^h_T$ of the optimal goal set $\mathcal{G}^*_T$ that is polynomial and requires no search at all. We will call $\mathcal{G}^h_T$, the *heuristic goal set*, and $\mathcal{G}^s_T$, the *suboptimal goal set*. The heuristic goal set $\mathcal{G}^h_T$ is defined as the set of goals $G \in \mathcal{G}$ for which $|\pi(G; s_0) \cap A_o|$ is highest, where $\pi(G; s_0)$ is the *relaxed plan* obtained for the goal G in P' from the initial state $s_0 = I$ of P'. In other words, while the computation of $\mathcal{G}^*_T$ involves an optimal planner, and the computation of the approximation $\mathcal{G}^s_T$ involves a suboptimal planner, the computation of the approximation $\mathcal{G}^h_T$ involves the computation of *one* relaxed plan and neither search nor planning. Below, we compare the accuracy and scalability of these different methods where we will see that the computation of the the sets $\mathcal{G}^s_T$ and $\mathcal{G}^h_T$ yields good approximations of the optimal goal set $\mathcal{G}^*_T$ and scale up much better.

For improving the quality of the approximations further, the implementation below accommodates two additional changes. The purpose of these changes is to allow 'gaps' in the explanation of the observations. In the description used up to now, if the observation sequence is $O = o_1, \ldots, o_m$, no plan or relaxed plan is able to account for the observation o_i if it doesn't account for all the observations o_k preceding o_i in the sequence. This restriction is used in the definition of the optimal goal set $\mathcal{G}^*_T$ and is needed in the methods that compute this set exactly. On the other hand, it imposes unnecessary limits on the approximation methods that are not required to account for all the observations. Clearly, a goal G_1 that explains all the observations o_i except for the first one o_1 should be selected over a possible goal G_2 that explains none, yet unless we allow suboptimal and relaxed plans to 'skip' observations, this is not possible. Thus, in the implementation, we allow 'relaxed plans' to skip observations by computing the set-additive heuristic and the relaxed plans using a further relaxation where the preconditions p_a that express the precedence constraints among the actions o_a that account for the observations, are dropped. Instead, during the computation of the heuristic, we set the heuristic value of an action o_a to infinity if its support includes an action o_b such that the observation a precedes the observation b in O. The precedence constraints, however, are preserved in the execution model used in FF for performing the state progressions. In such a phase though, when the planner selects to execute the action a instead of the action o_a that accounts for the observation a in O, the planner compares a with a new action $forgo(a)$ that is not considered anywhere else by the planner. Basically, while the action o_a is like a but with a precondition p_b if b comes right before a in O and postcondition p_a, the action $forgo(a)$ is like o_a but without the precondition p_b. The action $forgo(a)$ thus says to leave the observation $a \in O$ unaccounted for (as o_a will not make it then into the plan), but by adding p_a to the current state, it opens the possibility of accounting for actions d that follow a in the observation sequence. The planner chooses then between a and $forgo(a)$ as between any two actions: by comparing the heuristic value of the resulting states s' and by breaking ties in favor of the action that maximizes $|\pi(s') \cap A_o|$.

6 Example

As an illustration, we consider an example drawn from the familiar Blocks World, with six blocks: T, S, E, R, Y and A, such that the initial situation is

$$I = \{clear(S), on(S,T), on(T,A), on(A,R), on(R,E), on(E,Y), ontable(Y)\} .$$

The possible goals in $\mathcal{G}$ are towers that express words: 'year' (G_1), 'yeast' (G_2), and 'tray' (G_3). The following observation sequence is then obtained

$$O = \{unstack(T,A), unstack(R,E), pickup(T)\} .$$

that results in the following extra (fluents and) goals in the transformed theory T':

$$G_o = \{p_{unstack(T,A)}, p_{unstack(R,E)}, p_{pickup(T)}\} .$$

The exact method described in Section 5.1 proceeds by establishing that $c^*(G_1) = 14$ and $c^*(G_1 \cup G_o) > 14$, so that G_1 is ruled out as a possible goal. Then it determines that

$c^*(G_2) = 14$ and $c^*(G_2 \cup G_o) > 14$, so that G_2 is ruled out as well. Finally, it computes $c^*(G_3) = c^*(G_3 \cup G_o\} = 14$, and concludes that $\mathcal{G}_T^* = \{G_3\}$, and hence that G_3 is the only goal the agent may be trying to achieve. The first approximate method described in Section 5.2 computes plans $\pi(G_1), \pi(G_2), \pi(G_3)$ for each of the goals with the modified FF planner, and finds that $|A_o \cap \pi(G_1)| = |A_o \cap \pi(G_2)| = 2$ while $|A_o \cap \pi(G_3)| = 3$, so that the suboptimal and optimal goal sets coincide; namely, $\mathcal{G}_T^s = \{G_3\}$. On the other hand, in this example, the heuristic goal set $\mathcal{G}_T^h$ is weaker, not excluding any of the possible goals in $\mathcal{G}$. In the domains below, however, we will see that the heuristic approach can do very well too.

7 Empirical Evaluation

We have tested the optimal and approximate algorithms proposed on several PR theories defined from Planning domains used as benchmarks in the official planning competitions.[2] In the GRID–NAVIGATION domain the tasks can be summarized with the question "where is the agent going?" with the hypothetical goals being a subset of the nodes in the navigation graph. The graphs were built by randomly connecting chain–like graphs of fixed–length, connecting nodes so that every node was reachable from any other node. In IPC–GRID$^+$ the query is the same, and to the challenges posed by GRID–NAVIGATION, we added the fact that, fixed the set of possible destinations, we tossed the keys necessary to reach those destinations at a restricted set of places. This has the effect that (potentially) many observations are relevant to many different hypotheses.[3] For LOGISTICS the question is "where are the packages being transported to?". Here the number of valid plans to consider is not as large, but these plans have many actions in common. In BLOCK-WORDS the question is "which is the 'word' the agent is trying to build?". Here we built three dictionaries with 20 English words, not featuring a letter twice, and further constrained to use only 8 different letters. Here we have both a potential combinatorial explosion of possibilities as well as the problem of ambiguity: if many words use the letter 'E', their plans will share a considerable number of actions involving the block 'E'.

Table 1 summarizes the experimental results of the three computational methods over hundreds – exact numbers reported under domain name – of plan recognition problems $T = \langle P, \mathcal{G}, O\rangle$ drawn from these four domains, where P is a planning problem without goals, $\mathcal{G}$ is a set of possible goals, and O is an observation sequence. The problems were generated by considering for each domain, the combination of several 'incomplete' planning problems P, several goal sets $\mathcal{G}$, and many observation sequences O. For a given P and a given (hidden) goal G in $\mathcal{G}$, we generated several observation sequences O by randomly taking a certain percentage of the actions in an optimal plan for $P[G]$. A ratio of 10% means that the observations contain 10% of the actions in the optimal plan, while a ratio of 100% means that the observation sequence is the whole plan. In the tables, five ratios are considered, and for each P and G, 13 different samples of observations O were generated: one for 100%, three for each of the other ratios. For each of the four domains, each row expresses averages taken for each observation ratio while varying P and G.

Table 1 shows these averages: the number of goals in $\mathcal{G}$, the ratio and number of observations used, the length of the optimal plans, the total time to compute the optimal goal set $\mathcal{G}_T^*$ and its resulting size. It then includes the length of the suboptimal plans found, the total time to construct the suboptimal goal set $\mathcal{G}_T^s$, and its resulting size too. Last is the information about the second approximation $\mathcal{G}_T^h$: the total time to compute it and its average size. The columns headed by the letters FPR, AR, and FNR provide further information about the *quality* of the suboptimal and heuristic approximations $\mathcal{G}_T^s$ and $\mathcal{G}_T^h$. *False Positives* indicate cases where the approximate methods fail to exclude a possible goal, and *False Negatives*, when they fail to include a possible goal. More precisely, in the columns of the table the *False Positive Ratio (FPR)* is $|\mathcal{G}_T^s \setminus \mathcal{G}_T^*|/|\mathcal{G}_T^s \cup \mathcal{G}_T^*|$, the *False Negative Ratio (FNR)* is $|\mathcal{G}_T^* \setminus \mathcal{G}_T^s|/|\mathcal{G}_T^s \cup \mathcal{G}_T^*|$, and the *Agreement Ratio (AR),* shown in bold, is $|\mathcal{G}_T^s \cap \mathcal{G}_T^*|/|\mathcal{G}_T^s \cup \mathcal{G}_T^*|$. The ratios for the heuristic method are the same with $\mathcal{G}_T^h$ in place of $\mathcal{G}_T^s$.

From the table, we can see that the approximate methods are orders of magnitude faster than the optimal method, and that they manage to filter the goal hypotheses in a way that often coincides with the optimal account. Indeed, both the suboptimal and the heuristic method, that is polynomial and does no search at all, do very well in 3 of the domains, and slightly worse in the more challenging 'Blocks-word' domain, where they fail to rule out some of the hypotheses. The optimal algorithms make no error at all, as they are used to define and compute the optimal goal set exactly. They are displayed here nonetheless to provide a reference to the other algorithms and to illustrate the type of problems that they can effectively solve. For larger problems, approximate methods are the way to go, and we believe that further progress in heuristics and search algorithms with these plan recognition problems in mind, can potentially deliver more powerful and more accurate recognition algorithms.

8 Discussion

We have provided a crisp definition of the plan recognition task given a domain theory and have developed good approximation methods that use domain-independent planning techniques and are able to scale up. The proposed model is more flexible than library-based approaches and admits certain straightforward extensions such as dealing with the observation of fluents, handling actions that must be observed when executed, and partial orders on the observations. A critical aspect of the model, however, is that it does not weight the possible hypotheses (goals) but it just filters them. A natural extension of the model involves attaching a weight $\Delta(G)$ to each possible goal given by the *difference* between the cost $c^*_{P'}(G')$ of solving the goals $G' = G \cup G_0$ of the transformed domain P', and the cost $c^*_{P'}(G)$ of solving the goal G alone.

[2] The pages for the different competitions can be obtained from `http://ipc.informatik.uni-freiburg.de`

[3] This domain is a modification of the one in IPC-1 where the agent is not limited to carry a single key at any time.

				Optimal			Suboptimal				Heuristic		
Domain	$\|\mathcal{G}\|$	%Obs	$\|O\|$	Length	Time	$\|\mathcal{G}_T^*\|$	Length	Time	FPR/AR/FNR(%)	$\|\mathcal{G}_T^s\|$	Time	FPR/AR/FNR(%)	$\|\mathcal{G}_T^h\|$
block-words (778)	20.3	10	1.1	8.6	1192.4	10.3	12.1	10.67	32 / **66.1** / 1.9	14.38	0.33	7.3 / **76** / 16.8	7.81
		30	2.9		1247.06	3.71	12.9	24.25	33.2 / **58.6** / 8.1	5.71	0.34	16.2 / **73.6** / 10.2	3.97
		50	4.3		1288.33	2.08	11.3	23.7	21.4 / **72.6** / 6	3.11	0.35	10.8 / **83.8** / 5.4	2.73
		70	6.4		1353.36	1.38	10.0	10.37	7.8 / **87.6** / 4.6	1.55	0.37	10 / **88.6** / 1.5	1.89
		100	8.6		1415.97	1.24	9.2	8.05	3.7 / **96.3** / 0	1.46	0.39	4.9 / **94** / 1.1	1.51
grid-navigation (718)	11.3	10	2.2	16.0	2351.86	2.03	16.9	0.89	7.7 / **91.1** / 1.2	2.83	0.52	6.3 / **93.2** / 0.4	2.60
		30	5.4		2119.61	1.25	16.7	0.94	0 / **100** / 0	1.26	0.55	0 / **100** / 0	1.23
		50	8.5		2052.87	1.14	16.6	0.99	0 / **100** / 0	1.15	0.56	0 / **100** / 0	1.15
		70	12		2075.93	1.04	16.6	1.05	0 / **100** / 0	1.05	0.58	0 / **100** / 0	1.05
		100	16.0		1914.86	1.03	16.0	1.15	0 / **100** / 0	1.02	0.63	0 / **100** / 0	1.02
ipc-grid+ (303)	6.5	10	1.7	13.5	124.55	2.01	13.6	0.65	4.4 / **94.8** / 0.7	2.2	0.1	5.7 / **93.6** / 0.7	2.27
		30	4.2		131.18	1.23	13.5	1.22	1.4 / **96.5** / 2.1	1.22	0.1	2.3 / **97.7** / 0	1.29
		50	6.8		135.61	1.21	13.8	0.17	0 / **98.9** / 1.1	1.19	0.1	0 / **100** / 0	1.21
		70	9.6		142.05	1.15	13.8	0.17	0 / **100** / 0	1.15	0.11	0.7 / **99.3** / 0	1.16
		100	13.5		149.31	1.12	13.7	0.2	0 / **100** / 0	1.12	0.11	0 / **100** / 0	1.12
logistics (648)	10.0	10	2	18.7	1432.86	2.45	19.1	0.49	10.9 / **84.8** / 4.3	2.8	0.25	21.8 / **77.9** / 0.3	3.81
		30	5.9		1853.51	1.27	18.8	0.53	8.3 / **85.7** / 6	1.3	0.26	11.1 / **88.9** / 0	1.61
		50	9.5		997.19	1.07	18.7	0.61	8.1 / **87.1** / 4.9	1.14	0.27	7.6 / **91.1** / 1.2	1.32
		70	13.4		365.16	1.02	18.7	0.63	6.1 / **90.5** / 3.4	1.07	0.29	5.6 / **93.5** / 0.8	1.13
		100	18.7		1179.3	1.0	18.7	0.7	6 / **90.2** / 3.7	1.08	0.32	2.4 / **96.3** / 1.2	1.04

Table 1: Comparison of optimal and approximate plan recognition methods. Figures shown are all averages over many problems as explained in the text: size of $\mathcal{G}$, ratio and number of observations, length of optimal plans for goals in $\mathcal{G}_T^*$, total time to compute $\mathcal{G}_T^*$ and resulting size; length of suboptimal plans for goals in $\mathcal{G}_T^s$, time to compute $\mathcal{G}_T^s$, accuracy ratios for suboptimal method (AR in bold), and size of set; total time to compute heuristic approximation $\mathcal{G}_T^h$, accuracy ratios for heuristic method, and size of set. See text for details.

Indeed, such weight $\Delta(G)$ provides a measure of how far the agent has to move away from the best plans for achieving G in order to account for the observations: the greater the distance, the less probable the hypothesis G given the observations. Indeed, such weights can be used to define a probability distribution over the goals given the observation, something that has not been easy to do over the space of all plans. In this work, we have considered the special case where the hypotheses G selected are the most likely with $\Delta(G) = 0$.

Acknowledgments

H. Geffner thanks the members of the NLP-Planning Seminar at the University of Edinburgh for interesting discussions about NLP, Planning, and Plan Recognition. We thank P. Haslum for making available HSP optimal planner code and documentation. This work is partially supported by grant TIN2006-15387-C03-03 from MEC/Spain.

References

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

[Charniak and Goldman, 1993] E. Charniak and R. P. Goldman. A bayesian model of plan recognition. *Artificial Intelligence*, 64:53–79, 1993.

[Cohen *et al.*, 1981] P. R. Cohen, C. R. Perrault, and J. F. Allen. Beyond question answering. In W. Lehnert and M. Ringle, editors, *Strategies for Natural Language Processing*. Lawrence Erlbaum Associates, 1981.

[Goldman *et al.*, 1999] R. P. Goldman, C. W. Geib, and C. A. Miller. A new model of plan recognition. In *Proceedings of the 1999 Conference on Uncertainty in Artificial Intelligence*, 1999.

[Haslum and Geffner, 2000] P. Haslum and H. Geffner. Admissible heuristics for optimal planning. In *Proc. of the Fifth International Conference on AI Planning Systems (AIPS-2000)*, pages 70–82, 2000.

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Hoffmann and Nebel, 2001] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[Kautz and Allen, 1986] H. Kautz and J. F. Allen. Generalized plan recognition. In *AAAI*, pages 32–37, 1986.

[Keyder and Geffner, 2008] E. Keyder and H. Geffner. Heuristics for planning with action costs revisited. In *Proceedings 18th European Conference on Artificial Intelligence (ECAI-08)*, 2008.

[Lekavý and Návrat, 2007] M. Lekavý and P. Návrat. Expressivity of strips-like and htn-like planning. In *Agent and Multi-Agent Systems: Technologies and Applications, Proc. of 1st KES Int. Symp.KES-AMSTA 2007*, pages 121–130, 2007.

[Lesh and Etzioni, 1995] N. Lesh and O. Etzioni. A sound and fast goal recognizer. In *Proc. IJCAI-95*, pages 1704–1710, 1995.

[Pentney *et al.*, 2006] W. Pentney, A. Popescu, S. Wang, H. Kautz, and M. Philipose. Sensor-based understanding of daily life via large-scale use of common sense. In *Proceedings of AAAI*, 2006.

[Schmidt *et al.*, 1978] C. Schmidt, N. Sridharan, and J. Goodson. The plan recognition problem: an intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11:45–83, 1978.

[Vilain, 1990] M. Vilain. Getting serious about parsing plans: A grammatical analysis of plan recognition. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 190–197, 1990.