Landmark-Based Plan Recognition

Ramon Fraga Pereira\textsuperscript{2} and Felipe Meneguzzi \textsuperscript{2}

Abstract. Recognition of goals and plans using incomplete evidence from action execution can be done efficiently by using planning techniques. In many applications it is important to recognize goals and plans not only accurately, but also quickly. In this paper, we develop a heuristic approach for recognizing plans based on planning techniques that rely on ordering constraints to filter candidate goals from observations. These ordering constraints are called landmarks in the planning literature, which are facts or actions that cannot be avoided to achieve a goal. We show the applicability of planning landmarks in two settings: first, we use it directly to develop a heuristic-based plan recognition approach; second, we refine an existing planning-based plan recognition approach by pre-filtering its candidate goals. Our empirical evaluation shows that our approach is not only substantially more accurate than the state-of-the-art in all available datasets, it is also an order of magnitude faster.

1 Introduction

As more computer systems require reasoning about what agents (both human and artificial) other than themselves are doing, the ability to accurately and efficiently recognize goals and plans from agent behavior becomes increasingly important. Plan recognition is the task of recognizing goals and plans based on often incomplete observations that include actions executed by agents and properties of agent behavior in an environment \cite{18}. Accurate plan recognition is important to monitor and anticipate agent behavior, such as in crime detection and prevention, monitoring activities, and elderly care. Most plan recognition approaches \cite{7,11} employ plan libraries (i.e., a library with all plans for achieving a set of goals) to represent potential agent behavior, resulting in approaches to recognize plans that are analogous to language parsing. Recent work \cite{16,15,13} use planning domain definitions (domain theories) to represent potential agent behavior, bringing plan recognition closer to planning algorithms. These approaches allow techniques used in planning algorithms to be employed for recognizing goals and plans.

In this paper, we develop a plan recognition approach that relies on planning landmarks \cite{14,12} to filter candidate goals and plans from the observations. Landmarks are properties (or actions) that every plan must satisfy (or execute) at some point in every plan execution to achieve a goal. Whereas in planning algorithms these landmarks are used to focus search, in our approach, they allow plan recognition algorithms to rule out candidate goals whose landmarks are missing from observations. Thus, based on planning landmarks, we develop an algorithm to filter candidate goals by estimating how many landmarks required by every goal in the set of candidate goals have been achieved within the observed actions. Since computing a subset of landmarks for a set of goals can be done very quickly, our approach can provide substantial runtime gains. In this way, we use this filtering algorithm in two settings. First, we build a landmark-based plan recognition heuristic that analyzes the amount of achieved landmarks to estimate the percentage of completion of each filtered candidate goal. Second, we show that the filter we develop can also be applied to other planning-based plan recognition approaches, such as the approach from Ramírez and Geffner \cite{16}.

We evaluate empirically our plan recognition approach against the current state-of-the-art \cite{16} by using openly available datasets for plan recognition developed by Ramírez and Geffner in \cite{16,15}, and which have been used to evaluate recent approaches to plan recognition \cite{5}. This dataset provides several domains and problems in which it is not trivial to recognize the intended goal from a set of candidate goals and observations. Using this dataset, we show that our approach has at least three advantages over existing approaches. First, by relaxing the filter using a small threshold our landmark-based plan recognition approach is more accurate than the current state-of-the-art \cite{16}. Second, our approach also provides substantially faster recognition time on its own and when used to improve existing plan recognition approaches. Finally, we show that our filtering algorithm provides substantial improvements in recognition time when used to improve existing plan recognition approaches.

This paper is organized as follows. Section 2 provides background on planning and plan recognition. In Section 3, we describe how we extract useful information from planning domain definitions. Sections 4 and 5 develop the key parts of our approach for plan recognition. We empirically evaluate our approach in Section 6, which shows the results of the experiments. In Section 7, we survey related work and compare the state of the art with our approach. Finally, in Section 8, we conclude this paper by discussing limitations, advantages and future directions of our approach.

2 Background

In this section, we provide essential background on planning terminology, and how we define plan recognition problems over planning domain definitions.

2.1 Planning

Planning is the problem of finding a sequence of actions (i.e., plan) that achieves a particular goal from an initial state. In this work, we adopt the terminology from Ghallab et al. \cite{8} to represent planning domains and problems. First, we define a state in the environment by the following Definition 1.

Definition 1 (Predicates and State). A predicate is denoted by an \( n \)-ary predicate symbol \( p \) applied to a sequence of zero or more terms \( \bar{x} \).
(τ₁, τ₂, ..., τₙ) – terms are either constants or variables. A state is a finite set of grounded predicates (facts) that represent logical values according to some interpretation. Facts are divided into two types: positive and negated facts, as well as constants for truth (∨) and falsehood (⊥).

**Definition 2 (Operator).** An operator \( a \) is represented by a triple \( \langle \text{name}(a), \text{pre}(a), \text{eff}(a) \rangle \); \( \text{name}(a) \) represents the description or signature of \( a \); \( \text{pre}(a) \) describes the preconditions of \( a \), a set of predicates that must exist in the current state for \( a \) to be executed; \( \text{eff}(a) \) represents the effects of \( a \). These effects are divided into \( \text{eff}(a)^+ \) (i.e., an add-list of positive predicates) and \( \text{eff}(a)^- \) (i.e., a delete-list of negated predicates).

A plain domain contains operator definitions, which represents the environment dynamics that guide an agent’s search for plans to achieve its goals. Operator definitions are used in the construction of a planning domain, which represents the environment dynamics that guide an agent’s search for plans to achieve its goals. An agent can modify the current state by executing actions according to Definition 3.

**Definition 3 (Action).** An action is a ground operator instantiated over its free variables. Thus, if all operator free variables are substituted by objects when instantiating an operator, we have an action.

**Definition 4 (Planning Domain).** A planning domain definition \( \Xi \) is represented by a pair \( (\Sigma, \Pi) \), which specifies the knowledge of the domain, and consists of a finite set of facts \( \Sigma \) and a finite set of actions \( \Pi \).

A planning instance, comprises both a planning domain and the elements of a planning problem, describing the initial state of the environment and the goal which an agent wishes to achieve as formalized in Definition 5.

**Definition 5 (Planning Instance).** A planning instance \( \Pi \) is represented by a triple \( (\Xi, I, G) \), in which \( \Xi = (\Sigma, \Pi) \) is the domain definition; \( I \subseteq \Sigma \) is the initial state specification, which is defined by specifying the value for all facts in the initial state; and \( G \subseteq \Sigma \) is the goal state specification, which represents a desired state to be reached.

Classical planning representations often separate the definition of \( I \) and \( G \) as part of a planning problem (to be used together within a domain \( \Xi \)). Finally, a plan is the solution to a planning problem, as formalized in Definition 6.

**Definition 6 (Plan).** A plan \( \pi \) for a plan instance \( \Pi = (\Xi, I, G) \) is a sequence of actions \( a₁, a₂, ..., aₙ \) that modifies the initial state \( I \) into one in which the goal state \( G \) holds by the successive execution of actions in a plan \( \pi \). A plan \( \pi^* \) with length \(|\pi^*|\) is optimal if there exists no other plan \( \pi' \) for \( \Pi \) such that \(|\pi'| < |\pi^*|\).

**2.2 Plan Recognition**

Plan recognition is the task of recognizing how agents achieve their goals by observing their interactions in an environment. In plan recognition, such observed interactions are defined as available evidence that can be used to recognize plans. Most plan recognition approaches require knowledge of an agent’s possible plans for representing its typical behavior. In other words, this knowledge provides the recipes (i.e., know-how) for achieving goals. These recipes are often called plan libraries and are used as input for many plan recognition approaches. However, in this work we use as input a planning domain definition, more specifically, we use the STRIPS fragment of PDDL. We follow Ramirez and Geffner to formally define a plan recognition problem over a planning domain definition as follows.

**Definition 7 (Plan Recognition Problem).** A plan recognition problem is a quadruple \( \Gamma_{PR} = (\Xi, I, G, O) \), in which \( \Xi = (\Sigma, \Pi) \) is the domain definition, and consists of a finite set of facts \( \Sigma \) and a finite set of actions \( \Pi \); \( I \) represents the initial state; \( G \) is the set of possible goals of a hidden goal \( G \), such that \( G \in G \); and \( O = \{o₁, o₂, ..., oₙ\} \) is an observation sequence of a plan execution with each observation \( o_i \in O \) being an action in the finite set of actions \( \Pi \) from the domain definition \( \Xi \). This observation sequence can be full or partial, which means that for a full observation sequence we observe all actions during the execution of an agent plan, and for a partial observation sequence, only a sub-sequence of actions of the execution of an agent plan is observed. The solution to this problem is to find a hidden goal \( G \) in the set of possible goals \( G \) that the observation sequence of a plan execution achieves.

**3 Extracting Recognition Information from Planning Definition**

In this section, we describe the process through which we extract useful information for plan recognition from a planning domain. First, we describe landmark extraction algorithms from the literature, and how we use these algorithms to our approach. Second, we show how we classify facts into partitions from planning action descriptions.

**3.1 Extracting Landmarks**

In the planning literature, landmarks are defined as necessary features that must be true at some point in every valid plan to achieve a particular goal. Landmarks are often partially ordered according to the sequence in which they must be achieved. Hoffman et al. define landmarks as follows.

**Definition 8 (Landmark).** Given a planning instance \( \Pi = (\Xi, I, G) \), a formula \( L \) is a landmark in \( \Pi \) iff \( L \) is true at some point along all valid plans that achieve \( G \) from \( I \). In other words, a landmark is a type of formula (e.g., conjunctive formula or disjunctive formula) over a set of facts that must be satisfied at some point along all valid plan executions.

For plan recognition problems, landmarks allow us to infer whether a sequence of observations cannot possibly lead to a certain goal. In order to extract the landmarks of a planning problem, we use two landmark extraction algorithms from the literature: 1) Hoffman et al. to extract conjunctive landmarks; and 2) Porteous and Cresswell to extract conjunctive and disjunctive landmarks. To represent landmarks and their ordering, these algorithms use a tree in which nodes represent landmarks and edges represent necessary prerequisites between landmarks. Each node in the tree represents a conjunction of facts that must be true simultaneously at some point during plan execution, and the root node is a landmark representing the goal state. These algorithms use a Relaxed Planning Graph (RPG), which is a leveled graph that ignores the delete-list effects of all actions, and this way, there are no mutex relations in this graph. Once the RPG is built, the algorithm extracts landmark candidates by back-chaining from the RPG level in which all facts of the goal state...
it is not possible to extract any fact partitions. However, it is possible to extract fact partitions from the EASY-IPC-GRID\footnote{Easy-IPC-GRID domain consists of an agent that moves in a grid using keys to open locked locations.} domain, such as Strictly Activating and Unstable Activating facts. Here, we use fact partitions to obtain additional information on fact landmarks. For example, consider an Unstable Activating fact landmark $L_{ua}$, so that if $L_{ua}$ is deleted from the current state, then it cannot be re-achieved. We can trivially determine that goals for which this fact is a landmark are unreachable, because there is no available action that achieves $L_{ua}$ again.

4 Filtering Candidate Goals from Landmarks in Observations

Key to our approach to plan recognition is the ability to filter candidate goals based on the evidence of fact landmarks and partitioned facts in preconditions and effects of observed actions in a plan execution. We now present a filtering process that analyzes fact landmarks in preconditions and effects of observed actions, and selects goals, from a set of candidate goals, that have achieved most of their associated landmarks.

This filtering process is detailed in function FilterCandidateGoals of Algorithm 1, which takes as input a plan recognition problem $\mathcal{P}_{Pl}$, which is composed of a planning domain definition $\Xi$, an initial state $I$, a set of candidate goals $G$, a set of observed actions $O$, and a filtering threshold $\theta$. Our algorithm iterates over the set of candidate goals $G$, and, for each goal $g \in G$, it extracts and classifies fact landmarks and partitions for $G$ from the initial state $I$ (Lines 4 and 5). We then check whether the observed actions $O$ contain fact landmarks or partitioned facts in either their preconditions or effects. At this point, if any Strictly Activating facts for the candidate goal $G$ are not in initial state $I$, then the candidate goal $G$ is no longer achievable and we discard it (Line 6). Subsequently, we check for Unstable Activating and Strictly Terminal facts of goal $G$ in the preconditions and effects of the observed actions $O$, and if we find any, we discard the candidate goal $G$ (Line 11). If we observe no facts from partitions as evidence from the actions in $O$, we move on to checking landmarks of $G$ within the actions in $O$. If we observe any landmarks in the preconditions and positive effects of the observed actions (Line 13), we compute the percentage of achieved landmarks for goal $G$. As we deal with partial observations in a plan execution some executed actions may be missing from the observation, thus whenever we identify a fact landmark, we also infer that its predecessors have been achieved. For example, let us consider that the set of fact landmarks to achieve a goal from a state is represented by the following ordered facts: $(at A) \prec (at B) \prec (at C) \prec (at D)$, and we observe just one action during a plan execution, and this observed action contains the $(at C)$ fact landmark as an effect. Based on this action from a partial observation, we can infer that the predecessors of $(at C)$ have been achieved before the observation, and thus, we also include them as achieved landmarks. Given the number of achieved fact landmarks of $G$, we estimate the percentage of fact landmarks that the observed actions $O$ have achieved according to the ratio between the amount of achieved fact landmarks and the total amount of landmarks (Line 21). Finally, after analyzing all candidate goals in $G$, we return the goals with the highest percentage of achieved landmarks within our filtering threshold $\theta$ (Line 23).
Algorithm 1 Filter candidate goals.

\textbf{Input:} A set of filtered candidate goals $\Lambda$ with the highest percentage of achieved landmarks.

\textbf{Output:} A set of filtered candidate goals $\Lambda$ with the highest percentage of achieved landmarks.

\begin{algorithm}
\begin{algorithmic}[1]
\Function{FilterCandidateGoals}{$\Xi, I, \Gamma, O, \theta$}
\State $\Lambda := \emptyset$ \Comment{Map goals to $\%$ of landmarks achieved.}
\For{each goal $G$ in $\Gamma$}
\State $L_G := \text{ExtractLandmarks}(\Xi, I, \Gamma)$ \Comment{$F_{oa} = \text{PartitionFacts}(L_G)$}
\State $\text{Foa: Strictly Activating, } F_{ua}: \text{ Unstable Activating, } F_d: \text{ Strictly Terminal.}$
\If{$F_{oa} \cap I = \emptyset$} \Comment{Goal $G$ is no longer possible.}
\State continue
\EndIf
\If{$F_{ua}$} \Comment{Achieved landmarks for $G$.}
\For{each observed action $o \in O$}
\If{$(F_{oa} \cup F_{oa}) \subseteq (\text{pre}(o) \equiv \text{eff}(o)^+ \equiv \text{eff}(o)^-)$} \Comment{Avoid computing achieved landmarks for $G$.}
\State $\text{discard} = \text{true}$
\State break
\EndIf
\EndFor
\EndIf
\EndFor
\State $\Lambda_G := \Lambda_G \cup \langle \langle G, v \rangle, L \rangle$ \Comment{Percentage of achieved landmarks for $G$.}
\EndIf
\EndFor
\State return all $G$ s.t. $(\langle G, v \rangle, \Lambda_G) \in \Lambda_G$ and $v \geq \max_{v_i} \langle G_i, v_i \rangle - \theta$
\EndFunction
\end{algorithmic}
\end{algorithm}

us flexibility when dealing with incomplete observations and suboptimal plans, which, when $\theta = 0$, may cause potential goals to be filtered out before we get additional observations.

As an example of how the algorithm filters a set of candidate goals, consider the BLOCKS-WORLD example shown in Figure 1, which represents an initial configuration of stackable blocks, as well as set of candidate goals. The initial state is composed of sub-goals: atomic facts that are part of a conjunction of facts. This estimate represents the percentage of sub-goals (atomic facts that are part of a conjunction of facts) in a goal that have been accomplished based on the evidence of achieved fact landmarks in observations.

Our heuristic method estimates the percentage of completion towards a goal by using the set of achieved fact landmarks provided by the filtering process (Algorithm 1, Line 15). We aggregate the percentage of completion of each sub-goal into an overall percentage of completion for all facts in a candidate goal. This heuristic, denoted as $h_{prt}$, is computed by the formula below, where $\Lambda L_G$ is the number of achieved landmarks from observations of every sub-goal $g$ of the candidate goal $G$, and $\Lambda L_G$ represents the number of necessary landmarks to achieve every sub-goal $g$ of $G$:

\[ h_{prt}(G) = \left( \frac{\sum g \in G |\Lambda L_G|}{|G|} \right) \]

Thus, heuristic $h_{prt}(G)$ estimates the completion of a goal $G$ by calculating the ratio between the sum of the percentage of completion for every sub-goal $g \in G$, i.e., $\sum_{g \in G} |\Lambda L_G| / |G|$, and the number of sub-goals in $G$.

To exemplify how heuristic $h_{prt}$ estimates goal completion, recall the BLOCKS-WORLD example from Figure 1. For the BED goal its sub-goals (shown at the top of Figure 2) are: (clear B), (on B E), (on E D), and (on D). Based on the observed actions (stack D E) and (pick-up S), we conclude that sub-goals (clear B) and (on D) have already been achieved because they are in the initial state, and the observed actions do not delete any of those facts. Although fact (clear E) in the initial state does not correspond to the final configuration of goal BED, we account for this fact in the heuristic calculation, since we consider all observed evidence. At this point, our heuristic computes that 50% of goal BED
Figure 2: Fact landmarks for the word BED.

Figure 3: Fact landmarks for the word RED.

6 Landmark-based Plan Recognition

We now bring together the techniques from Sections 4 and 5 into our landmark-based plan recognition approach that uses the presented filter and heuristic for recognizing goals and plans. Our plan recognition approach is detailed in Algorithm 2. This algorithm takes as input a planning domain \( \mathcal{S} \), an initial state \( \mathcal{I} \), \( \mathcal{G} \) set of candidate goals, \( \mathcal{O} \) observations, and \( \theta \) threshold.

Algorithm 2 Recognize goals and plans using the filtering method and the landmark-based heuristic.

Input: \( \Xi = (\mathcal{S}, \mathcal{A}) \) planning domain, \( \mathcal{I} \) initial state, \( \mathcal{G} \) set of candidate goals, \( \mathcal{O} \) observations, and \( \theta \) threshold.

Output: Recognized goal(s).

1: function \( \text{RECOGNIZE}(\Xi, \mathcal{I}, \mathcal{G}, \mathcal{O}, \theta) \)
2: \( \Lambda_0 := \emptyset \) \> Map goals to % of landmarks achieved.
3: \( \Lambda_\theta := \text{FILTERCANDIDATEGOMALS}(\Xi, \mathcal{I}, \mathcal{G}, \mathcal{O}, \theta) \)
4: return \( \arg \max_{G \in \Lambda_\theta} h_{prl}(G) \)
5: end function

7 Experiments and Evaluation

In this section, we describe the experiments and evaluation we carried out on our landmark-based plan recognition approach against state-of-the-art techniques. For experiments, we use six domains from datasets provided by Ramírez and Geffner [16, 15], comprising hundreds of problems. We summarize these domains as follows.

- **BLOCKS-WORLD** domain consists of a set of blocks, a table, and a robot hand. Blocks can be stacked on top of other blocks or on the table. A block that has nothing on it is clear. The robot hand can hold one block or be empty. The goal is to find a sequence of actions that achieves a final configuration of blocks;

- **CAMPUS** domain consists of finding what activity is being performed by a student from his observations on campus environment;

- **EASY-IPC-GRID** domain consists of an agent that moves in a grid from connected cells to others by transporting keys in order to open locked locations;

- **INTRUSION-DETECTION** represents a domain where a hacker tries to access, vandalize, steal information, or perform a combination of these attacks on a set of servers;

- **KITCHEN** is a domain that consists of home-activities, in which the goals can be preparing dinner, breakfast, among others; and

- **LOGISTICS** is a domain which models cities, and each city contains locations. These locations are airports. For transporting packages between locations, there are trucks and airplanes. Trucks can drive between cities. Airplanes can fly between airports. The goal is to get and transport packages from locations to other locations.

These domains contain hundreds of plan recognition problems, i.e., a domain description as well as an initial state, a set of candidate goals \( G \), a hidden goal \( G \) in \( G \), and an observation sequence \( O \). An observation sequence contains actions that represent an optimal plan or sub-optimal plan that achieves a hidden goal \( G \), and this observation sequence can be full or partial. A full observation sequence represents the whole plan for a hidden goal \( G \), i.e., 100% of the actions having been observed. A partial observation sequence represents a plan for a hidden goal \( G \) with 10%, 30%, 50%, or 70% of its actions having been observed. Our experiments use two metrics, the accuracy of the recognition and the speed to recognize a goal. We compare our approach to two other approaches: the approach of Ramírez and Geffner [15], more specifically, we use their faster and most accurate approach; as well as a combination of their approach and our filter.

For evaluation, we use the accuracy metric (true positive rate), which represents how well a hidden goal is recognized from a set

6 https://goo.gl/qLF6wB
of possible goals for a given plan recognition problem; as well as recognition time (in seconds), which represents how long it takes for a hidden goal to be recognized given a plan recognition problem. In the Blocks-World domain, the accuracy metric measures how well these approaches recognize, from observations, the word that is being assembled. Regarding CAMPUS, the accuracy with which the approaches correctly infer the goal. For the experiments with the KITCHEN domain, we aim to accurately how well these approaches recognize the activity is being performed by an observed student. For the EASY-IPC-GRID domain, how accurate these approaches recognize the cell where keys are being transported by the observed agent. With regard to INTRUSION-DETECTION, how accurate these approaches recognize the type of attack and servers that are being hacked observations. In the KITCHEN domain, we aim to accurately how well these approaches recognize the activity being performed. For LOGISTICS, how accurate these approaches recognize the location where the packages are being transported from observations. Besides the accuracy metric, we use the Receiver Operating Characteristic (ROC), which is called ROC curve. ROC curve shows graphically the performance of a binary classifier system by evaluating true positive rate against the false positive rate at various threshold settings (in this paper we evaluate plan recognition approaches). More specifically, we use the ROC curve to compare not only true positive predictions (i.e., accuracy), but also to compare the false positive ratio of the experiments plan recognition approaches. Table 1 compares the results for the three plan recognition approaches, showing the total number of plan recognition problems under each domain name. For each domain we show the number of candidate goals |G| and varying percentages of the plan that is actually observed, as well as the average number of observed actions per problem |O|. Note that, for partial observations, random observed actions are removed (up to the set percentage), but the order is maintained. |L| denotes the average number of fact landmarks extracted for each domain. For each approach, we compute the time to recognize the hidden goal (seconds), given the observations, and the accuracy with which the approaches correctly infer the goal. For our landmark-based plan recognition approach, we show the accuracy under different filtering thresholds (0%, 10%, 20%, and 30%). If threshold θ = 0, our approach does not give any flexibility for filtering.
ing candidate goals, returning only the goals with the highest percentage of achieved landmarks. Each row of this table shows the observability (% Obs) and averages of the number of candidate goals |G|, the number of observed actions |O|, recognition time, and accuracy. From this table, it is possible to see that our landmark-based plan recognition approach is both faster and more accurate than Ramírez and Geffner [16], and, when we combine their algorithm with our filter, the resulting approach gets a substantial speedup. Importantly, as we increase the threshold, our plan recognition approach quickly surpasses the state of the art in all domains tested. We note that when measuring time to recognition using our filter we also include the time to compute landmarks, so that landmark computation is performed online (i.e., during the process of plan recognition). Thus, even if this computation has a complex upper bound, in our experience, computing landmarks (especially conjunctive ones) is very fast.

Table 1 shows that both our landmark-based plan recognition approach and Ramírez and Geffner’s [16] yield near perfect accuracy for recognizing goals and plans for all planning domains. However, by using the ROC curve we highlight the trade-off between true positive results and false positive results for these plan recognition approaches. Figures 6-9 show the ROC curve for the six planning domains we use. In the ROC curve, the diagonal line in (Random Access) represents a random guess to recognize a goal from observations. This diagonal line divides the ROC space, in which points above the diagonal represent good classification results (better than random), whereas points below the line represent poor results (worse than random). The best possible (perfect) prediction for recognizing goals must be a point in the upper left corner (i.e., coordinate x = 0 and y = 100) in the ROC space. Thus, the closer a plan recognition approach (point) gets to the upper left corner, the better it is for recognizing goals and plans. Blue, green, red, and yellow points with five different symbols represent our plan recognition approach varying the use of the threshold (0%, 10%, 20% and 30%). These five different symbols represent the percentage of observability (10%, 30%, 50%, 70% and 100%) with regard to the observed plan. Black points represent Ramírez and Geffner’s [16] approach (R&G). According to the ROC curve in Figures 6-9 we see that all variation (using different thresholds) of our plan landmark-based recognition approach yield good (sometimes perfect) predictions for recognizing goals and plans, in contrast to R&G, which is near-perfect in these four domains. Figure 10 shows that the results for the BLOCKS-WORLD are quite scattered in the ROC curve, so recognizing goals and plans in this domain is difficult. Nevertheless, it possible to see
that our plan recognition is not only competitive (using the thresholds between 10% and 20%) with R&G with superior accuracy, but also at least 8.75 orders of magnitude faster than R&G. Finally, Figure 5 shows that in the EASY-IPC-GRID domain our approach is very competitive with R&G, again with consistently higher accuracy, but also is near perfect for false positives, surpassing R&G by using different thresholds.

8 Related Work
Ramírez and Geffner [16] propose planning approaches for plan recognition, and instead of using plan-libraries, they model the problem as a planning domain theory with respect to a known set of goals. Their work uses a heuristic, an optimal and modified sub-optimal planner to determine the distance to every goal in a set of goals after an observation. Follow-up work [13] proposes a probabilistic plan recognition approach using off-the-shelf planners. These approaches yield high accuracy in most domains, however, this accuracy is lower than in our threshold-based approaches, and their time to recognition ranges from twice slower to up to an order of magnitude slower. Pat- tison and Long [13] propose IGRAPH (AUTomatic Goal Recogni- tion with A Planning Heuristic), a probabilistic heuristic-based goal recognition over planning domains. IGRAPH uses heuristic estimation and domain analysis to determine which goals an agent is pursu- ing. Although they adapt their features, their problem definition is formally different than ours, preventing direct comparison. In [11], Keren et al. present an alternate view regarding the goal and plan recognition problem. This work uses planning techniques to assist in the design of goal and plan recognition problems. Most recently, E.-Martín et al. [5] propose a planning-based plan recognition approach that propagates cost and interaction information in a plan graph, and uses this information to estimate goal probabilities over the set of candidate goals. Although our landmark-based plan recognition approach has no probabilistic interpretation, the accuracy of our approach seems to be higher in the same domains.

9 Conclusion
We have developed an approach for plan recognition that relies on planning landmarks and a new heuristic based on these landmarks. Landmarks provide key information about what cannot be avoided to achieve a goal, and we show that landmarks can be used efficiently for very accurate plan recognition. We have shown empirically that our approach yields not only superior accuracy results but also substantially faster recognition times for all domains used in evaluating against the state of the art [10] at varying observation completeness levels.

Our experiments show that in at least one domain, disjointive landmarks have a positive effect on accuracy with minimal effect on recognition time, whereas in some domains, these landmarks are either not present or yield almost no gain in accuracy at substantial loss of speed. Knowledge of the domains leads us to believe that disjointive landmarks are most useful in domains in which we assume that observed plans are just sub-optimal, such as the KITCHEN domain. Conversely, disjointive landmarks slow down recognition in domains in which there are multiple mutually exclusive plans towards the same goal, such as the EASY-IPC-GRID domain in which the agent moves in a grid.

We intend to explore multiple avenues for future work. First, we aim to evaluate other planning techniques, such as heuristics and symmetries in classical planning [17]. Second, we intend to explore other landmark extraction algorithms to obtain additional information from planning domains, such as temporal landmarks [10]. Third, we aim to model a probability interpretation to the observed landmarks and compare the probability results (and probabilistic accuracy) to the recent work of E.-Martin et al. [5]. Finally, for domains with goals with intersecting landmarks, we can use measures of information gain to weigh observations to help break ties when multiple goals are left after the filter. Given the computational complexity of landmark extraction in the general case, we aim to theoretically analyze the tradeoff between landmark completeness and runtime ef- ficiency.

REFERENCES