Landmark-Based Heuristics for Goal Recognition

Ramon Fraga Pereira,† Nir Oren,‡ Felipe Meneguzzi†
†Pontifical Catholic University of Rio Grande do Sul, Brazil
ramon.pereira@acad.pucrs.br
felipe.meneguzzi@pucrs.br
‡University of Aberdeen, United Kingdom
n.oren@abdn.ac.uk

Abstract

Automated planning can be used to efficiently recognize goals and plans from partial or full observed action sequences. In this paper, we propose goal recognition heuristics that rely on information from planning landmarks — facts or actions that must occur if a plan is to achieve a goal when starting from some initial state. We develop two such heuristics: the first estimates goal completion by considering the ratio between achieved and extracted landmarks of a candidate goal, while the second takes into account how unique each landmark is among landmarks for all candidate goals. We empirically evaluate these heuristics over both standard goal/plan recognition problems, and a set of very large problems. We show that our heuristics can recognize goals more accurately, and run orders of magnitude faster, than the current state-of-the-art.

1 Introduction

Goal recognition is the task of recognizing agents' goals from partial or full observed action sequences. Most goal and plan recognition approaches (Avrahami-Zilberbrand and Kaminka 2005; Geib and Goldman 2009; Mirsky et al. 2016) employ plan libraries to represent agent behavior (*i.e.*, a library that describes all plans for achieving goals), and plan recognition techniques which use such libraries are analogous to parsing. However, work that use classical planning domain definitions to represent potential agent behavior bring goal and plan recognition closer to automated planning (Ramírez and Geffner 2009; Ramírez and Geffner 2010; Pattison and Long 2010; Keren, Gal, and Karpas 2014; E.-Martín, R.-Moreno, and Smith 2015; Sohrabi, Riabov, and Udrea 2016; Pereira and Meneguzzi 2016). These approaches — which do not use plan libraries — show that automated planning techniques can be used to efficiently recognize goals and plans. In many domains where goal and plan recognition are important (*e.g.*, smart environments, user monitoring and crime detection), plan libraries may be unavailable, making this second class of approaches important. In this paper, we propose two goal recognition heuristics based on planning techniques that rely on planning landmarks (Hoffmann, Porteous, and Sebastia 2004). In automated planning, landmarks are properties (or actions) that every plan must satisfy (or execute) at some point in every plan execution to achieve a goal. Although landmarks are often used to build planning algorithms (Richter and Westphal 2010), our approach allows goal recognition heuristics to rule out candidate goals whose landmarks are inconsistent with the observations.

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Our contribution is twofold. First, we develop a new heuristic approach for goal recognition that obviates the need to execute a planner multiple times (Ramírez and Geffner 2009) yielding substantial runtime gains. Second, we develop two goal recognition heuristics which exploit landmarks. The first of our heuristics estimates goal completion by considering the ratio between achieved and extracted landmarks of a candidate goal. The second heuristic uses a *landmark uniqueness value*, representing the information value of the landmark for some specific candidate goal when compared to landmarks for all candidate goals. This latter heuristic then estimates which candidate goal is correct by summing the uniqueness value of the achieved landmarks from the observations. We empirically evaluate our goal recognition heuristics against the current state-of-the-art (2009) by using a dataset developed by Ramírez and Geffner (2009; 2010), and a dataset we generated for other planning domains with larger problems. We show that our heuristics are substantially faster and more accurate than the state-of-the-art for datasets that contain several domains and problems where recognizing the intended goal is non-trivial.

2 Background

2.1 Planning

Planning is the problem of finding a sequence of actions that achieves a goal from an initial state (Ghallab, Nau, and Traverso 2004). A *state* is a finite set of facts that represent logical values according to some interpretation. *Facts* can be either positive, or negated ground predicates. A predicate is denoted by an n-ary predicate symbol p applied to a sequence of zero or more terms (τ_1, τ_2, ..., τ_n). An *operator* is represented by a triple $a = \langle \mathit{name}(a), \mathit{pre}(a), \mathit{eff}(a)\rangle$ where *name*(a) represents the description or signature of a; *pre*(a) describes the preconditions of a — a set of facts or predicates that must exist in the current state for a to be executed; $\mathit{eff}(a) = \mathit{eff}(a)^+ \cup \mathit{eff}(a)^-$ represents the effects of a, with $\mathit{eff}(a)^+$ an *add-list* of positive facts or predicates, and *eff*$(a)^-$ a *delete-list* of negative facts or predicates. When we instantiate an operator over its free variables, we call the resulting ground operator an *action*. A *planning instance* is represented by a triple $\Pi = \langle \Xi, \mathcal{I}, G \rangle$, in which $\Xi = \langle \Sigma, \mathcal{A} \rangle$ is a *planning domain definition*; Σ consists of a finite set of facts and $\mathcal{A}$ a finite set of actions; $\mathcal{I} \subseteq \Sigma$ is the initial state; and $G \subseteq \Sigma$ is the goal state. A *plan* is a sequence of actions $\pi = \langle a_1, a_2, ..., a_n \rangle$ that modifies the initial state $\mathcal{I}$ into one in which the goal state G holds by the successive execution of actions in a plan π. While actions have an associated cost, as in classical planning, we assume that this cost is 1 for all actions. A plan π is considered optimal if its cost, and thus length, is minimal.

2.2 Goal Recognition

Goal recognition is the task of recognizing agents' goals by observing their interactions in an environment (Sukthankar et al. 2014). In goal recognition, such observed interactions are defined as available evidence that can be used to recognize goals. As proposed by Ramírez and Geffner (2009; 2010), we formally define a goal recognition problem over a planning domain definition as follows.

Definition 1 (**Goal Recognition Problem**). *A goal recognition problem is a tuple $T_{GR} = \langle \Xi, \mathcal{I}, \mathcal{G}, O \rangle$, in which $\Xi = \langle \Sigma, \mathcal{A} \rangle$ is a planning domain definition; $\mathcal{I}$ is the initial state; $\mathcal{G}$ is the set of possible goals, which include a hidden goal G (i.e., $G \in \mathcal{G}$); and $O = \langle o_1, o_2, ..., o_n \rangle$ is an observation sequence of executed actions, with each observation $o_i \in \mathcal{A}$.*

The solution for a goal recognition problem is the hidden goal $G \in \mathcal{G}$ that the observation sequence O of a plan execution achieves. Note that the observation sequence can be full or partial — in a full observation sequence we observe all actions of an agent's plan; in a partial observation sequence, only a sub-sequence of actions are observed.

2.3 Landmarks

In the planning literature, landmarks are defined as necessary properties (actions) that must be true (executed) at some point in every valid plan to achieve a particular goal. Landmarks are often partially ordered based on the sequence in which they must be achieved. Hoffman *et al.* (2004) define landmarks as follows:

Definition 2 (**Fact Landmark**). *Given a planning instance $\Pi = \langle \Xi, \mathcal{I}, G \rangle$, a formula L is a fact landmark in Π iff L is true at some point along all valid plans that achieve G from $\mathcal{I}$. A landmark is a type of formula (e.g., a conjunctive or disjunctive formula) over a set of facts that must be satisfied at some point along all valid plan executions.*

Landmark extraction algorithms such as that of Hoffman *et al.* (2004) can extract conjunctive landmarks from a planning instance. To represent landmarks and their ordering, this algorithm uses 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. Hoffman *et al.* (2004) proves that the process of generating all landmarks and deciding their ordering is PSPACE-complete, which is exactly the same complexity as deciding plan existence (Bylander 1994). Thus, to operate efficiently, most landmark extraction algorithms extract only a subset of landmarks for a given planning instance.

For our approach, we use Hoffman *et al.* (2004)'s algorithm, which extracts a set of fact landmarks for every goal G in the set of candidate goals $\mathcal{G}$ from the initial $\mathcal{I}$. We refer to this algorithm as the function EXTRACTLANDMARKS, which takes as input a planning domain definition $\Xi = \langle \Sigma, \mathcal{A} \rangle$; an initial state $\mathcal{I}$; and a set of candidate goals. This function outputs a map $\mathcal{L}_{\mathcal{G}}$ that associates candidate goals to their respective ordered fact landmarks (*i.e.*, a set of landmarks with an order relation).

3 Computing Achieved Landmarks

Our approach requires the ability to identify which fact landmarks were achieved based on a partial sequence of action observations. The evidence we use to infer fact landmarks consist of the preconditions and effects of observed actions during plan execution. This is achieved by function COMPUTEACHIEVEDLANDMARKS shown in Algorithm 1, which takes as input an initial state $\mathcal{I}$, a set of candidate goals $\mathcal{G}$, a sequence of observed actions O, and a map $\mathcal{L}_{\mathcal{G}}$ containing candidate goals and their extracted fact landmarks. Given these input parameters, the algorithm iterates over the set of candidate goals $\mathcal{G}$ (Line 3) selecting the fact landmarks $\mathcal{L}_G$ of each goal G in $\mathcal{L}_{\mathcal{G}}$ in Line 4 and computes the fact landmarks that are in the initial state in Line 5. With this information, the algorithm iterates over the observed actions O to compute the achieved fact landmarks of G in Lines 6 to 9. For each observed action o in O, the algorithm computes all fact landmarks of G that are in the preconditions and effects of o in Line 7. As we deal with partial observations in a plan execution some executed actions may be missing from the observation sequence, thus whenever we identify a fact landmark, we also infer that its predecessors must have been achieved in Line 8. For example, 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 fact landmark `(at C)` as an effect. From this observed action, we can infer that the predecessors of `(at C)` must have been achieved before this observation (*i.e.*, `(at A)` and `(at B)`), and therefore, we also include them as achieved landmarks. At the end of each iteration over an observed action o, the algorithm stores the set of achieved landmarks of G in $\mathcal{AL}_G$ in Line 9. Finally, after computing the evidence of achieved landmarks in the observations for a candidate goal G, the algorithm stores the set of achieved landmarks $\mathcal{AL}_G$ of G in $\Lambda_{\mathcal{G}}$ (Line 11) and returns a map $\Lambda_{\mathcal{G}}$ containing all candidate goals and their respective achieved fact landmarks (Line 13).

As an example of how the algorithm computes achieved fact landmarks from observations, let us consider the BLOCKS-WORLD example shown in Figure 1, which shows an initial state and a set of candidate goals. The initial state

Algorithm 1 Compute Achieved Landmarks From Observations.

Input: *$\mathcal{I}$ initial state, $\mathcal{G}$ set of candidate goals, O observations*, and *$\mathcal{L}_\mathcal{G}$ goals and their extracted landmarks.*
Output: *A map of goals to their achieved landmarks.*

```
 1: function ComputeAchievedLandmarks($\mathcal{I}, \mathcal{G}, O, \mathcal{L}_\mathcal{G}$)
 2:     $\Lambda_\mathcal{G} \leftarrow \langle\rangle$        ▷ Map goals $\mathcal{G}$ to their respective achieved landmarks.
 3:     for each goal $G$ in $\mathcal{G}$ do
 4:         $\mathcal{L}_G \leftarrow$ fact landmarks of $G$ s.t $\langle G, \mathcal{L}_G \rangle$ in $\mathcal{L}_\mathcal{G}$
 5:         $\mathcal{L}_\mathcal{I} \leftarrow$ all fact landmarks $L \in \mathcal{I}$
 6:         for each observed action $o$ in $O$ do
 7:             $\mathcal{L} \leftarrow$ all fact landmarks $L$ in $\mathcal{L}_G$ such that $L \in pre(o) \cup eff(o)^+$ and $L \notin \mathcal{L}$
 8:             $\mathcal{L}_\prec \leftarrow$ predecessors $L_\prec$ of all $L$ in $\mathcal{L}$, such that $L_\prec \notin \mathcal{L}$
 9:             $\mathcal{AL}_G \leftarrow \mathcal{AL}_G \cup \{\mathcal{L}_\mathcal{I} \cup \mathcal{L} \cup \mathcal{L}_\prec\}$
10:         end for
11:         $\Lambda_\mathcal{G}(G) \leftarrow \mathcal{AL}_G$        ▷ Achieved landmarks of G.
12:     end for
13:     return $\Lambda_\mathcal{G}$
14: end function
```

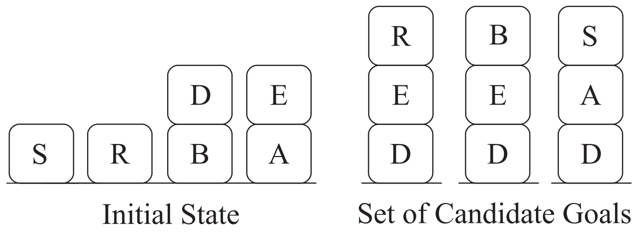

Figure 1: Blocks-World example.

represents an initial configuration of stackable blocks, while the set of candidate goals is composed by the following stackable words: RED, BED, and SAD. As observed actions, assume we have: (unstack E A) and (stack E D). For this example, we take the candidate goal RED to show how the algorithm computes achieved landmarks. Thus, from these observed actions, the candidate goal RED, and the set of fact landmarks of this candidate goal (Figure 2), our algorithm computes that the following fact landmarks have been achieved:

- $\mathcal{AL}_{\text{RED}} = \{$[(clear R)], [(on E D)], [(clear R) (ontable R) (handempty)], [(on E A) (clear E) (handempty)], [(clear D) (holding E)], [(on D B) (clear D) (handempty)]$\}$

In the preconditions of (unstack E A) the algorithm computes [(on E A) (clear E) (handempty)]. Subsequently, in the preconditions and effects of (stack E D) the algorithm computes [(clear D) (holding E)] and [(on E D)], while it computes the other achieved landmarks for the word RED from the initial state. Figure 2 shows the set of achieved landmarks for the word RED in gray. Listing 1 shows in bold the set of achieved landmarks that our algorithm computes for the set of candidate goals in Figure 1.

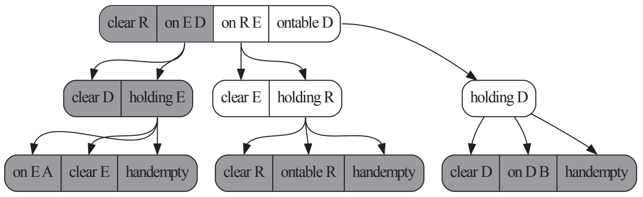

Figure 2: Ordered fact landmarks extracted for the stacked blocks RED. Fact landmarks that must be true together are represented by connected boxes. Connected boxes in grey represent achieved fact landmarks. Edges represent prerequisites between landmarks.

4 Landmark-Based Goal Completion Heuristic

We now describe a goal recognition heuristic which selects a goal from a set of possible candidates based on the number of landmarks that have been detected, and are required to achieve that goal (computed using ComputeAchievedLandmarks). We note that a candidate goal is composed of sub-goals comprised of the atomic facts that are part of a conjunction of facts.

Our heuristic operates by aggregating the percentage of completion of each sub-goal into an overall percentage of completion for all facts of a candidate goal. This heuristic, denoted as h_{gc}, is formally defined by Equation 1, where $\mathcal{AL}_g$ is the number of achieved landmarks from observations of every sub-goal g of the candidate goal G in $\mathcal{AL}_G$, and $\mathcal{L}_g$ represents the number of necessary landmarks to achieve every sub-goal g of G in $\mathcal{L}_G$.

$$h_{gc}(G, \mathcal{AL}_G, \mathcal{L}_G) = \left(\frac{\sum_{g \in G} \frac{|\mathcal{AL}_g \in \mathcal{AL}_G|}{|\mathcal{L}_g \in \mathcal{L}_G|}}{|G|} \right) \qquad (1)$$

Thus, heuristic $h_{gc}(G)$ 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.*, $\sum_{g \in G} \frac{|\mathcal{AL}_g \in \mathcal{AL}_G|}{|\mathcal{L}_g \in \mathcal{L}_G|}$, and the size $|G|$ of the set of sub-goals, that is, the number of sub-goals in G. Algorithm 2 describes how to recognize goals using the h_{gc} heuristic and takes as input a goal recognition problem, as well as a threshold value θ. The θ threshold gives us flexibility to avoid eliminating candidate goals whose the percentage of goal completion are close to the highest completion value. In Line 2, the algorithm uses the ExtractLandmarks function to extract landmarks for all candidate goals. From the initial state $\mathcal{I}$, the observations O, and the extracted landmarks $\mathcal{L}_\mathcal{G}$, in Line 3, our algorithm computes the set of achieved landmarks $\Lambda_\mathcal{G}$ for every candidate goal using Algorithm 1. Finally, the algorithm uses the heuristic h_{gc} to estimate goal completion for every candidate G in $\mathcal{G}$, and as output (Line 5), the algorithm returns those candidate goals with the highest estimated value within the threshold θ.

As an example of how heuristic h_{gc} estimates goal completion, recall the Blocks-World example from Figure 1. Consider that among these candidate goals the hid-

Algorithm 2 Recognize goals/plans using the heuristic h_{gc}.

Input: Ξ *planning domain definition*, $\mathcal{I}$ *initial state*, $\mathcal{G}$ *set of candidate goals*, O *observations*, and θ *threshold*.
Output: *Recognized goal(s).*

```
1: function RECOGNIZE($\Xi, \mathcal{I}, \mathcal{G}, O, \theta$)
2:     $\mathcal{L}_{\mathcal{G}} \leftarrow$ EXTRACTLANDMARKS($\Xi, \mathcal{I}, \mathcal{G}$)
3:     $\Lambda_{\mathcal{G}} \leftarrow$ COMPUTEACHIEVEDLANDMARKS($\mathcal{I}, \mathcal{G}, O, \mathcal{L}_{\mathcal{G}}$)
4:     $maxh \leftarrow \max_{G' \in \mathcal{G}} h_{gc}(G', \Lambda_{\mathcal{G}}(G'), \mathcal{L}_{\mathcal{G}}(G'))$
5:     return all $G$ s.t $G \in \mathcal{G}$ and
           $h_{gc}(G, \Lambda_{\mathcal{G}}(G), \mathcal{L}_{\mathcal{G}}(G)) \geq (maxh - \theta)$
6: end function
```

den correct goal is RED, and we observe the following partial sequence of actions that achieve RED: (unstack E A) and (stack E D). Thus, based on the achieved landmarks $\mathcal{AL}_{\text{RED}}$ computed using Algorithm 3 (Figure 2), our heuristic h_{gc} estimates that the percentage of completion for the goal RED is 0.66: (clear R) = $1/1$ + (on E D) = $3/3$ + (on R E) = $1/3$ + (ontable D) = $1/3$, and hence, $2.66/4$ = 0.66. For the words BED and SAD our heuristic h_{gc} estimates respectively, 0.54 and 0.58.

5 Landmark-Based Uniqueness Heuristic

Many goal recognition problems containing multiple ambiguous candidate goals have these goals sharing common fact landmarks. Clearly, landmarks that are common to multiple goals are less useful for recognizing a goal than landmarks that exist for only a single goal. As a consequence, computing how unique (and thus informative) each landmark is can help disambiguate similar goals for a set of candidate goals. We develop a second heuristic building on this intuition; to construct this heuristic, we introduce the concept of *landmark uniqueness*, which is the inverse frequency of a landmark among the landmarks found in a set of candidate goals. For example, consider a landmark L that occurs only for a single goal within a set of candidate goals; the uniqueness value for such a landmark is intuitively the maximum value of 1. Equation 2 formalizes this intuition, describing how the *landmark uniqueness value* is computed for a landmark L and a set of landmarks for goals $\mathcal{L}_{\mathcal{G}}$.

Using this uniqueness value, we estimate which candidate goal is the intended one by summing the uniqueness values of the landmarks achieved in the observations. Unlike our previous heuristic, which estimates progress towards goal completion by analyzing sub-goals and their achieved landmarks, the landmark-based uniqueness heuristic estimates the goal completion of a candidate goal G by calculating the ratio between the sum of the uniqueness value of the achieved landmarks of G and the sum of the uniqueness value of all landmarks of G. Effectively, this algorithm weighs the completion value by the informational value of a landmark so that unique landmarks have the highest weight. To estimate goal completion using the landmark uniqueness value, we must calculate the uniqueness value for every extracted landmark in the set of landmarks of the candidate goals. Using Equation 2, we compute the landmark uniqueness value of every landmark L of $\mathcal{L}_{\mathcal{G}}$ and store it into Υ_{uv}. This heuristic is denoted as h_{uniq} and formally defined in Equation 3.

$$L_{Uniq}(L, \mathcal{L}_{\mathcal{G}}) = \left(\frac{1}{\sum_{\mathcal{L} \in \mathcal{L}_{\mathcal{G}}} |\{L | L \in \mathcal{L}\}|} \right) \tag{2}$$

$$h_{uniq}(G, \mathcal{AL}_G, \mathcal{L}_G, \Upsilon_{uv}) = \left(\frac{\sum_{\mathcal{A}_L \in \mathcal{AL}_G} \Upsilon_{uv}(\mathcal{A}_L)}{\sum_{L \in \mathcal{L}_G} \Upsilon_{uv}(L)} \right) \tag{3}$$

Algorithm 3 formalizes a goal recognition function that uses the h_{uniq} heuristic. This algorithm takes as input the same parameters as the previous approach: a goal recognition problem and a threshold θ. Like Algorithm 1, this algorithm extracts the set of landmarks for all candidate goals from the initial state $\mathcal{I}$, stores them in $\mathcal{L}_{\mathcal{G}}$ (Line 2), and computes the set of achieved landmarks based on the observations, storing these in $\Lambda_{\mathcal{G}}$. Unlike Algorithm 1, in Line 6 this algorithm computes the landmark uniqueness value for every landmark L in $\mathcal{L}_{\mathcal{G}}$ and stores it into Υ_{uv}. Finally, using these computed structures, the algorithm recognizes which candidate goal is being pursued from observations using the heuristic h_{uniq}, returning those candidate goals with the highest estimated value within the θ threshold.

Algorithm 3 Recognize goals/plans using the heuristic h_{uniq}.

Input: Ξ *planning domain definition*, $\mathcal{I}$ *initial state*, $\mathcal{G}$ *set of candidate goals*, O *observations*, and θ *threshold*.
Output: *Recognized goal(s).*

```
1: function RECOGNIZE($\Xi, \mathcal{I}, \mathcal{G}, O, \theta$)
2:     $\mathcal{L}_{\mathcal{G}} \leftarrow$ EXTRACTLANDMARKS($\Xi, \mathcal{I}, \mathcal{G}$)
3:     $\Lambda_{\mathcal{G}} \leftarrow$ COMPUTEACHIEVEDLANDMARKS($\mathcal{I}, \mathcal{G}, O, \mathcal{L}_{\mathcal{G}}$)
4:     $\Upsilon_{uv} \leftarrow \langle\rangle$          ▷ Map of landmarks to their uniqueness value.
5:     for each fact landmark $L$ in $\mathcal{L}_{\mathcal{G}}$ do
6:         $\Upsilon_{uv}(L) \leftarrow L_{Uniq}(L, \mathcal{L}_{\mathcal{G}})$
7:     end for
8:     $maxh \leftarrow \max_{G' \in \mathcal{G}} h_{uniq}(G', \Lambda_{\mathcal{G}}(G'), \mathcal{L}_{\mathcal{G}}(G'), \Upsilon_{uv})$
9:     return all $G$ s.t $G \in \mathcal{G}$ and
           $h_{uniq}(G, \Lambda_{\mathcal{G}}(G), \mathcal{L}_{\mathcal{G}}(G), \Upsilon_{uv}) \geq (maxh - \theta)$
10: end function
```

As an example of how our heuristic h_{uniq} estimates goal completion using landmark uniqueness values, recall the BLOCKS-WORLD example from Figure 1. As previously, the correct hidden goal is RED and we observe the following actions: (unstack E A) and (stack E D). Listing 1 shows the set of extracted fact landmarks for the candidate goals in the BLOCKS-WORLD example and their respective uniqueness value. Based on the set of achieved landmarks (shown in bold in Listing 1), our heuristic h_{uniq} estimates the following percentage for each candidate goal: h_{uniq}(RED) = $3.66/6.33$ = 0.57; h_{uniq}(BED) = $1.66/6.33$ = 0.27; and h_{uniq}(SAD) = $3.66/8.33$ = 0.43. In this case, Algorithm 3 correctly estimates RED to be the intended goal since it has the highest heuristic value.

```
- (and (clear B) (on B E) (on E D) (ontable D)) = 6.33
  [(on E D)] = 0.5, [(clear D) (holding E)] = 0.5,
  [(on E A) (clear E) (handempty)] = 0.33, [(ontable D)] = 0.33,
  [(on D B) (clear D) (handempty)] = 0.33, [(holding D)] = 0.33,
  [(clear B)] = 1.0, [(clear E) (holding B)] = 1.0,
  [(on B E)] = 1.0, [(clear B) (ontable B) (handempty)] = 1.0,

- (and (clear S) (on S A) (on A D) (ontable D)) = 8.33
  [(clear S)] = 1.0, [(on A D)] = 1.0, [(on S A)] = 1.0,
  [(clear A) (ontable A) (handempty)] = 1.0, [(ontable D)] = 0.33,
  [(clear S) (ontable S) (handempty)] = 1.0, [(holding D)] = 0.33,
  [(on E A) (clear E) (handempty)] = 0.33,
  [(on D B) (clear D) (handempty)] = 0.33,
  [(clear A) (holding S)] = 1.0, [(clear D) (holding A)] = 1.0

- (and (clear R) (on R E) (on E D) (ontable D)) = 6.33
  [(clear R)] = 1.0, [(clear R) (ontable R) (handempty)] = 1.0,
  [(clear D) (holding E)] = 0.5, [(on E D)] = 0.5,
  [(on E A) (clear E) (handempty)] = 0.33, [(ontable D)] = 0.33,
  [(on D B) (clear D) (handempty)] = 0.33, [(holding D)] = 0.33,
  [(on R E)] = 1.0, [(clear E) (holding R)] = 1.0
```

Listing 1: Extracted fact landmarks for the BLOCKS-WORLD example and their respective uniqueness value.

6 Experiments and Evaluation

We empirically evaluate our approach using datasets created using 15 domains from the planning literature[1]. Six of these datasets are also used in the evaluation of Ramírez and Geffner (2009; 2010). In addition, we generate new datasets from the remaining nine planning domains (using open-source planners, such as FAST-DOWNWARD, FAST-FORWARD, and LAMA), each of which is based on planning problems containing both optimal and sub-optimal plans of various sizes, including large problems to test the scalability of the approach. These domains contain hundreds of non-trivial goal/plan recognition problems, *i.e.*, a domain description as well as an initial state, a set of candidate goals $\mathcal{G}$, a hidden goal G in $\mathcal{G}$, 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 partial or full. While the latter capture all actions used to achieve hidden goal G (*i.e.*, with 100% observed actions), partial observation sequences represent plans for G where 10%, 30%, 50% or 70% of actions are observed.

Our evaluation uses two metrics, namely accuracy of goal recognition and goal recognition speed. In this paper, we measure accuracy by counting the number of times an algorithm includes the hidden goal in the set of candidate goals generated by the algorithm. Note that in many domains, the number of candidate goals returned by all algorithms may be more than one. In the case of Ramírez and Geffner (2009), this may occur when goals have the same distance from their estimated state. Alternatively, for our heuristics this may occur when there are ties between the heuristic value of candidate goals within the threshold margin. We compare our heuristic approaches to the fastest and most accurate approach from Ramírez and Geffner (2009). To visualize the comparative performance of the multiple approaches we adapt the notation of the Receiver Operating Characteristic (ROC) curve, and, rather than plotting a single curve per domain, we aggregate multiple domains and plot these results in ROC space. ROC space graphically shows the performance of a binary classifier system by evaluating the true positive rate against the false positive rate at various threshold settings. More specifically, we use ROC space to compare not only true positive predictions (*i.e.*, accuracy), but also to compare the false positive ratio of the evaluated goal/plan recognition approaches.

[1]http://ipc.icaps-conference.org

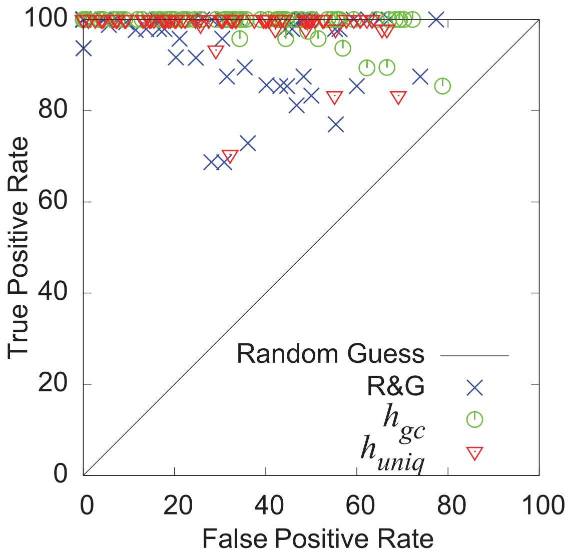

Figure 3: Comparative performance in ROC space.

Table 1 compares the results of our heuristics h_{gc} and h_{uniq} against the Ramírez and Geffner's approach (2009) (R&G in the table), by showing the total number of goal recognition problems used under each domain name. Each row of this table expresses averages for the number of candidate goals $|\mathcal{G}|$; the average number of extracted landmarks $|\mathcal{L}|$; the percentage of the plan that is actually observed; the average number of observations per problem $|O|$; and for each approach, the time in seconds to recognize the goal given the observations; and the accuracy with which the approaches correctly infer the goal. From this table, we can see that our landmark-based heuristics are both faster and more accurate than R&G (2009), and as we increase the threshold θ, our heuristics quickly surpass the state-of-the-art for all domains tested by yielding near perfect accuracy. We note that when measuring recognition time, we also include the time to extract the set of landmarks, so that landmark extraction is performed online, *i.e.*, during the process of goal recognition.

Figure 3 shows the trade-off between true positive results and false positive results for the tested goal/plan recognition approaches. In the ROC curve, the diagonal line represents a random guess to recognize a goal from observations, *i.e.*, 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 $y = 100$)

					h_{gc}		h_{uniq}		R&G	
Domain	$\|\mathcal{G}\|$	$\|\mathcal{L}\|$	**%Obs**	$\|O\|$	**Time** θ **(0 / 10 / 20 / 30)**	**Accuracy** θ **(0 / 10 / 20 / 30)**	**Time** θ **(0 / 10 / 20 / 30)**	**Accuracy** θ **(0 / 10 / 20 / 30)**	**Time**	**Accuracy**
BLOCKS-WORLD (855)	20	15.6	10	1.1	0.099 / 0.100 / 0.105 / 0.111	36.9% / 37.4% / 70.2% / 89.2%	0.096 / 0.098 / 0.101 / 0.105	31.9% / 37.4% / 59.4% / 70.4%	1.656	83.8%
			30	2.9	0.107 / 0.109 / 0.118 / 0.122	54.4% / 61.5% / 86.1% / 97.4%	0.103 / 0.106 / 0.107 / 0.111	56.6% / 62.5% / 82.5% / 93.3%	1.735	90.0%
			50	4.2	0.113 / 0.113 / 0.120 / 0.127	62.5% / 82.5% / 98.3% / 100.0%	0.108 / 0.109 / 0.112 / 0.115	70.4% / 86.1% / 93.3% / 98.9%	1.836	97.2%
			70	6.5	0.138 / 0.139 / 0.141 / 0.148	83.5% / 94.8% / 100.0% / 100.0%	0.118 / 0.121 / 0.125 / 0.129	76.6% / 86.1% / 96.9% / 100.0%	2.056	98.8%
			100	8.5	0.163 / 0.166 / 0.172 / 0.185	100.0% / 100.0% / 100.0% / 100.0%	0.136 / 0.142 / 0.146 / 0.151	100.0% / 100.0% / 100.0% / 100.0%	2.378	100.0%
CAMPUS (75)	2	8.5	10	1	0.038 / 0.039 / 0.042 / 0.044	93.3% / 100.0% / 100.0% / 100.0%	0.034 / 0.035 / 0.038 / 0.039	100.0% / 100.0% / 100.0% / 100.0%	0.083	100.0%
			30	2	0.048 / 0.050 / 0.055 / 0.057	100.0% / 100.0% / 100.0% / 100.0%	0.042 / 0.043 / 0.044 / 0.046	100.0% / 100.0% / 100.0% / 100.0%	0.091	100.0%
			50	3	0.063 / 0.062 / 0.066 / 0.068	93.3% / 100.0% / 100.0% / 100.0%	0.054 / 0.055 / 0.056 / 0.058	93.3% / 100.0% / 100.0% / 100.0%	0.105	100.0%
			70	4.4	0.060 / 0.060 / 0.063 / 0.065	100.0% / 100.0% / 100.0% / 100.0%	0.057 / 0.058 / 0.059 / 0.061	100.0% / 100.0% / 100.0% / 100.0%	0.112	100.0%
			100	5.5	0.068 / 0.069 / 0.073 / 0.072	100.0% / 100.0% / 100.0% / 100.0%	0.062 / 0.063 / 0.065 / 0.064	100.0% / 100.0% / 100.0% / 100.0%	0.126	100.0%
DEPOTS (208)	8.5	26.5	10	2.8	0.841 / 0.933 / 1.084 / 1.166	41.6% / 56.2% / 79.1% / 85.4%	0.799 / 0.886 / 0.923 / 1.017	39.5% / 50.0% / 70.8% / 79.1%	3.293	85.4%
			30	8	0.928 / 1.187 / 1.307 / 1.404	60.4% / 75.0% / 89.5% / 93.7%	0.865 / 1.051 / 1.009 / 1.114	54.1% / 68.7% / 83.3% / 91.6%	4.760	87.5%
			50	12.9	1.053 / 1.379 / 1.422 / 1.518	87.5% / 87.5% / 95.8% / 95.8%	1.019 / 1.136 / 1.122 / 1.218	85.4% / 87.5% / 95.8% / 97.9%	6.302	91.6%
			70	18	1.262 / 1.488 / 1.504 / 1.594	89.5% / 89.5% / 100.0% / 100.0%	1.140 / 1.205 / 1.214 / 1.333	89.5% / 89.5% / 100.0% / 100.0%	9.156	93.7%
			100	25.2	1.397 / 1.556 / 1.598 / 1.681	100.0% / 100.0% / 100.0% / 100.0%	1.228 / 1.312 / 1.347 / 1.425	100.0% / 100.0% / 100.0% / 100.0%	15.228	93.7%
DRIVER-LOG (208)	6.5	8.5	10	2	0.203 / 0.216 / 0.224 / 0.228	39.5% / 50.0% / 72.9% / 89.5%	0.152 / 0.156 / 0.164 / 0.178	37.5% / 52.0% / 70.8% / 83.3%	1.054	100.0%
			30	5.4	0.229 / 0.233 / 0.239 / 0.244	56.2% / 72.9% / 81.2% / 95.8%	0.159 / 0.163 / 0.179 / 0.184	50.0% / 72.9% / 81.2% / 97.9%	1.346	100.0%
			50	8.6	0.238 / 0.241 / 0.256 / 0.260	77.0% / 81.2% / 93.7% / 95.8%	0.168 / 0.171 / 0.186 / 0.190	70.0% / 81.2% / 91.6% / 97.9%	1.792	100.0%
			70	12	0.247 / 0.250 / 0.263 / 0.266	85.4% / 87.5% / 100.0% / 100.0%	0.177 / 0.180 / 0.195 / 0.206	89.5% / 91.6% / 100.0% / 100.0%	2.353	100.0%
			100	16.5	0.256 / 0.269 / 0.271 / 0.288	100.0% / 100.0% / 100.0% / 100.0%	0.185 / 0.194 / 0.201 / 0.213	100.0% / 100.0% / 100.0% / 100.0%	3.265	100.0%
DOCK-WORKER-ROBOTS (208)	6.75	40.5	10	5	0.797 / 0.856 / 0.901 / 0.988	41.6% / 70.8% / 89.5% / 100.0%	0.688 / 0.754 / 0.803 / 0.861	50.0% / 68.7% / 81.2% / 97.9%	2.805	87.5%
			30	13.8	0.886 / 0.919 / 0.977 / 1.046	66.6% / 93.7% / 100.0% / 100.0%	0.773 / 0.855 / 0.890 / 0.979	64.5% / 87.5% / 95.8% / 100.0%	4.171	83.3%
			50	22.6	0.931 / 0.998 / 1.093 / 1.160	72.9% / 97.9% / 100.0% / 100.0%	0.791 / 0.933 / 1.093 / 1.116	66.6% / 93.7% / 100.0% / 100.0%	6.154	72.9%
			70	31.7	1.063 / 1.103 / 1.225 / 1.322	97.9% / 100.0% / 100.0% / 100.0%	0.875 / 1.001 / 1.073 / 1.221	93.7% / 100.0% / 100.0% / 100.0%	13.973	68.7%
			100	44.6	1.125 / 1.197 / 1.324 / 1.448	100.0% / 100.0% / 100.0% / 100.0%	0.987 / 1.105 / 1.212 / 1.304	100.0% / 100.0% / 100.0% / 100.0%	35.306	68.7%
EASY-IPC-GRID (465)	7.5	11.3	10	1.8	0.585 / 0.588 / 0.609 / 0.623	82.2% / 85.5% / 97.7% / 100.0%	0.378 / 0.391 / 0.406 / 0.454	65.5% / 82.2% / 93.3% / 100.0%	1.206	97.7%
			30	4.3	0.597 / 0.600 / 0.614 / 0.644	86.6% / 93.3% / 97.7% / 100.0%	0.384 / 0.402 / 0.415 / 0.471	91.1% / 96.6% / 96.6% / 100.0%	1.291	98.8%
			50	6.9	0.608 / 0.609 / 0.627 / 0.656	94.4% / 97.7% / 97.7% / 100.0%	0.409 / 0.411 / 0.472 / 0.500	96.6% / 98.8% / 100.0% / 100.0%	1.306	98.8%
			70	9.8	0.629 / 0.628 / 0.661 / 0.715	95.5% / 98.8% / 98.8% / 100.0%	0.427 / 0.440 / 0.494 / 0.522	98.8% / 100.0% / 100.0% / 100.0%	1.715	100.0%
			100	13.3	0.630 / 0.632 / 0.685 / 0.759	100.0% / 100.0% / 100.0% / 100.0%	0.445 / 0.474 / 0.518 / 0.573	100.0% / 100.0% / 100.0% / 100.0%	2.263	100.0%
FERRY (208)	7.25	26.5	10	2.6	0.104 / 0.113 / 0.128 / 0.136	64.5% / 91.6% / 100.0% / 100.0%	0.088 / 0.091 / 0.100 / 0.102	58.3% / 91.6% / 100.0% / 100.0%	0.605	97.9%
			30	7	0.112 / 0.119 / 0.133 / 0.139	89.5% / 97.9% / 100.0% / 100.0%	0.092 / 0.103 / 0.107 / 0.109	87.5% / 93.7% / 100.0% / 100.0%	0.955	100.0%
			50	11.2	0.125 / 0.128 / 0.135 / 0.144	93.7% / 100.0% / 100.0% / 100.0%	0.097 / 0.109 / 0.111 / 0.114	89.5% / 100.0% / 100.0% / 100.0%	1.187	100.0%
			70	15.7	0.129 / 0.131 / 0.136 / 0.147	100.0% / 100.0% / 100.0% / 100.0%	0.101 / 0.110 / 0.112 / 0.115	100.0% / 100.0% / 100.0% / 100.0%	1.676	97.9%
			100	22	0.137 / 0.145 / 0.149 / 0.158	100.0% / 100.0% / 100.0% / 100.0%	0.105 / 0.113 / 0.116 / 0.123	100.0% / 100.0% / 100.0% / 100.0%	2.598	100.0%
INTRUSION-DETECTION (465)	15	16	10	1.9	0.197 / 0.200 / 0.211 / 0.233	76.4% / 96.6% / 100.0% / 100.0%	0.140 / 0.147 / 0.152 / 0.166	67.7% / 100.0% / 100.0% / 100.0%	1.130	98.8%
			30	4.5	0.214 / 0.219 / 0.227 / 0.241	94.4% / 100.0% / 100.0% / 100.0%	0.148 / 0.159 / 0.165 / 0.174	84.4% / 100.0% / 100.0% / 100.0%	1.142	100.0%
			50	6.7	0.218 / 0.221 / 0.246 / 0.269	100.0% / 100.0% / 100.0% / 100.0%	0.155 / 0.168 / 0.173 / 0.182	100.0% / 100.0% / 100.0% / 100.0%	1.203	100.0%
			70	9.5	0.219 / 0.223 / 0.258 / 0.274	100.0% / 100.0% / 100.0% / 100.0%	0.161 / 0.172 / 0.184 / 0.199	100.0% / 100.0% / 100.0% / 100.0%	1.482	100.0%
			100	13.1	0.277 / 0.281 / 0.303 / 0.325	100.0% / 100.0% / 100.0% / 100.0%	0.184 / 0.200 / 0.221 / 0.247	100.0% / 100.0% / 100.0% / 100.0%	1.567	100.0%
KITCHEN (75)	3	5	10	1.3	0.003 / 0.003 / 0.002 / 0.004	93.3% / 100.0% / 100.0% / 100.0%	0.002 / 0.002 / 0.003 / 0.003	100.0% / 100.0% / 100.0% / 100.0%	0.099	100.0%
			30	3.5	0.003 / 0.004 / 0.005 / 0.005	93.3% / 100.0% / 100.0% / 100.0%	0.003 / 0.003 / 0.002 / 0.003	100.0% / 100.0% / 100.0% / 100.0%	0.111	100.0%
			50	4	0.004 / 0.004 / 0.006 / 0.006	93.3% / 100.0% / 100.0% / 100.0%	0.003 / 0.004 / 0.004 / 0.005	100.0% / 100.0% / 100.0% / 100.0%	0.112	100.0%
			70	5	0.006 / 0.007 / 0.007 / 0.008	93.3% / 93.3% / 100.0% / 100.0%	0.005 / 0.007 / 0.007 / 0.007	100.0% / 100.0% / 100.0% / 100.0%	0.111	100.0%
			100	7.4	0.007 / 0.008 / 0.008 / 0.009	100.0% / 100.0% / 100.0% / 100.0%	0.006 / 0.007 / 0.007 / 0.009	100.0% / 100.0% / 100.0% / 100.0%	0.118	100.0%
LOGISTICS (465)	10	18.7	10	2	0.441 / 0.449 / 0.455 / 0.458	73.3% / 96.6% / 100.0% / 100.0%	0.360 / 0.373 / 0.391 / 0.408	57.7% / 90.0% / 100.0% / 100.0%	1.125	100.0%
			30	5.9	0.447 / 0.452 / 0.461 / 0.466	88.7% / 100.0% / 100.0% / 100.0%	0.377 / 0.388 / 0.400 / 0.412	85.5% / 94.4% / 100.0% / 100.0%	1.195	100.0%
			50	9.5	0.457 / 0.469 / 0.474 / 0.488	96.6% / 100.0% / 100.0% / 100.0%	0.385 / 0.409 / 0.416 / 0.424	85.5% / 100.0% / 100.0% / 100.0%	1.248	98.8%
			70	13.4	0.474 / 0.481 / 0.490 / 0.497	100.0% / 100.0% / 100.0% / 100.0%	0.401 / 0.418 / 0.425 / 0.432	97.7% / 100.0% / 100.0% / 100.0%	1.507	100.0%
			100	18.7	0.498 / 0.505 / 0.513 / 0.522	100.0% / 100.0% / 100.0% / 100.0%	0.417 / 0.426 / 0.433 / 0.441	100.0% / 100.0% / 100.0% / 100.0%	1.984	100.0%
MICONIC (208)	6	18	10	2.2	0.151 / 0.156 / 0.162 / 0.175	58.3% / 97.9% / 100.0% / 100.0%	0.103 / 0.108 / 0.115 / 0.126	56.2% / 95.8% / 100.0% / 100.0%	0.725	100.0%
			30	6	0.158 / 0.160 / 0.163 / 0.181	95.8% / 100.0% / 100.0% / 100.0%	0.109 / 0.116 / 0.121 / 0.130	87.5% / 100.0% / 100.0% / 100.0%	1.107	100.0%
			50	9.5	0.154 / 0.165 / 0.177 / 0.184	95.8% / 100.0% / 100.0% / 100.0%	0.112 / 0.127 / 0.133 / 0.141	93.7% / 100.0% / 100.0% / 100.0%	1.664	100.0%
			70	13.4	0.163 / 0.174 / 0.186 / 0.192	100.0% / 100.0% / 100.0% / 100.0%	0.124 / 0.136 / 0.147 / 0.156	100.0% / 100.0% / 100.0% / 100.0%	2.131	100.0%
			100	18.5	0.179 / 0.185 / 0.193 / 0.201	100.0% / 100.0% / 100.0% / 100.0%	0.138 / 0.143 / 0.155 / 0.167	100.0% / 100.0% / 100.0% / 100.0%	3.098	100.0%
ROVERS (208)	6	14.6	10	1.7	0.174 / 0.176 / 0.182 / 0.185	54.1% / 91.6% / 100.0% / 100.0%	0.143 / 0.152 / 0.163 / 0.173	56.2% / 85.4% / 97.9% / 100.0%	0.582	100.0%
			30	4	0.188 / 0.188 / 0.190 / 0.194	85.4% / 95.8% / 100.0% / 100.0%	0.154 / 0.167 / 0.174 / 0.188	89.5% / 95.8% / 100.0% / 100.0%	1.077	97.9%
			50	6.2	0.193 / 0.195 / 0.209 / 0.211	87.5% / 100.0% / 100.0% / 100.0%	0.165 / 0.170 / 0.185 / 0.193	89.5% / 97.9% / 100.0% / 100.0%	1.318	100.0%
			70	8.7	0.202 / 0.210 / 0.221 / 0.222	93.7% / 97.9% / 100.0% / 100.0%	0.177 / 0.181 / 0.194 / 0.200	97.9% / 97.9% / 100.0% / 100.0%	1.716	100.0%
			100	11.7	0.208 / 0.214 / 0.227 / 0.231	100.0% / 100.0% / 100.0% / 100.0%	0.182 / 0.193 / 0.201 / 0.208	100.0% / 100.0% / 100.0% / 100.0%	2.095	100.0%
SATELLITE (208)	6	14.6	10	1.5	0.462 / 0.464 / 0.469 / 0.485	41.6% / 68.7% / 100.0% / 100.0%	0.427 / 0.435 / 0.448 / 0.454	39.5% / 79.1% / 95.8% / 100.0%	0.812	100.0%
			30	4.1	0.465 / 0.471 / 0.476 / 0.491	58.3% / 83.3% / 97.9% / 100.0%	0.431 / 0.444 / 0.455 / 0.462	56.2% / 81.2% / 97.9% / 100.0%	1.361	97.9%
			50	6.5	0.470 / 0.476 / 0.480 / 0.494	79.1% / 93.7% / 100.0% / 100.0%	0.440 / 0.458 / 0.463 / 0.470	70.8% / 93.7% / 100.0% / 100.0%	1.564	100.0%
			70	9.3	0.477 / 0.481 / 0.488 / 0.502	95.8% / 100.0% / 100.0% / 100.0%	0.452 / 0.463 / 0.477 / 0.480	95.8% / 100.0% / 100.0% / 100.0%	1.855	95.8%
			100	12.5	0.484 / 0.490 / 0.495 / 0.508	100.0% / 100.0% / 100.0% / 100.0%	0.463 / 0.472 / 0.481 / 0.493	100.0% / 100.0% / 100.0% / 100.0%	2.318	100.0%
SOKOBAN (208)	8	8.25	10	2.3	1.020 / 1.025 / 1.034 / 1.046	62.5% / 83.3% / 87.5% / 100.0%	0.915 / 0.926 / 0.939 / 0.944	52.0% / 72.9% / 85.4% / 97.9%	4.136	77.0%
			30	6.7	1.031 / 1.032 / 1.046 / 1.058	66.6% / 83.3% / 87.5% / 100.0%	0.923 / 0.941 / 0.947 / 0.955	62.5% / 83.3% / 85.4% / 100.0%	7.775	87.5%
			50	10.6	1.043 / 1.048 / 1.050 / 1.061	77.0% / 91.6% / 97.9% / 100.0%	0.936 / 0.945 / 0.952 / 0.960	70.8% / 89.5% / 95.8% / 100.0%	11.179	85.4%
			70	15	1.059 / 1.060 / 1.067 / 1.070	79.1% / 91.6% / 97.9% / 100.0%	0.947 / 0.954 / 0.963 / 0.975	83.3% / 91.6% / 95.8% / 100.0%	17.026	85.4%
			100	20.8	1.066 / 1.067 / 1.073 / 1.079	100.0% / 100.0% / 100.0% / 100.0%	0.955 / 0.962 / 0.971 / 0.982	100.0% / 100.0% / 100.0% / 100.0%	25.217	81.2%
ZENO-TRAVEL (208)	7.5	7	10	1.9	1.231 / 1.235 / 1.234 / 1.239	43.7% / 68.7% / 91.6% / 100.0%	1.122 / 1.131 / 1.148 / 1.157	39.5% / 54.1% / 89.5% / 97.9%	2.063	97.9%
			30	4.4	1.234 / 1.237 / 1.239 / 1.248	72.9% / 83.3% / 91.6% / 100.0%	1.137 / 1.144 / 1.155 / 1.163	75.0% / 79.1% / 91.6% / 100.0%	4.182	89.5%
			50	6.8	1.238 / 1.242 / 1.245 / 1.251	81.2% / 93.7% / 95.8% / 100.0%	1.145 / 1.156 / 1.168 / 1.174	87.5% / 91.6% / 93.7% / 100.0%	6.157	95.8%
			70	9.7	1.243 / 1.244 / 1.252 / 1.264	93.7% / 93.7% / 100.0% / 100.0%	1.154 / 1.165 / 1.173 / 1.185	91.6% / 93.7% / 100.0% / 100.0%	8.307	97.9%
			100	13.3	1.252 / 1.251 / 1.260 / 1.272	100.0% / 100.0% / 100.0% / 100.0%	1.166 / 1.173 / 1.184 / 1.191	100.0% / 100.0% / 100.0% / 100.0%	10.851	100.0%

Table 1: Comparison of accuracy and recognition time against Ramírez and Geffner's approach (2009).

in the ROC space. The closer a goal recognition approach (point) is to the upper left corner, the better it is for recognizing goals and plans. To compare our recognition results against R&G in the ROC curve, we select the results of our heuristics using the $\theta = 30\%$ threshold. For each approach, we plot its recognition results for all domains into a cloud of points, which represents (in general) how well each approach recognizes the correct goal from observations. The points in ROC space show that our heuristics are not only competitive with R&G for all variations of observability, but also surpass R&G in a substantial number of domains.

Finally, we compare the time that each approach takes to recognize the hidden goal for different sizes of the observation sequence. We illustrate runtime in Figure 4, which summarizes, for the three evaluated approaches, the runtime (Time columns in Table 1) as a function of the average size of the observations ($|O|$ column in Table 1). Curves in the graph were generated by averaging the runtime when observation sizes were the same and smoothing over the resulting points. The graph shows the scalability of the three evaluated approaches. Our heuristics never take more than two seconds to compute the hidden goal in the set of candidate goals, while R&G's approach appears to grow superlinearly. As shown for the DOCK-WORKER-ROBOTS and SOKOBAN domains, larger plan lengths also lead R&G's approach to lose accuracy.

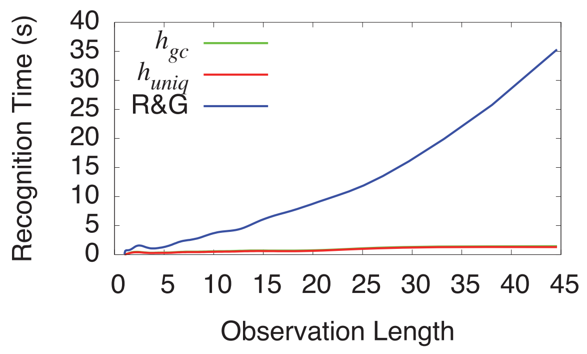

Figure 4: Recognition time comparison.

7 Related Work

Pattison and Long (2010) propose AUTOGRAPH (AUTOmatic Goal Recognition with A Planning Heuristic), a probabilistic heuristic-based goal recognition over planning domains. AUTOGRAPH uses heuristic estimation and domain analysis to determine which goals an agent is pursuing. Ramírez and Geffner (2009) 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. In this paper, we compare their most accurate approach directly with ours. Follow-up work (2010) proposes a probabilistic plan recognition approach using off-the-shelf planners. E.-Martín *et al.* (2015) propose a planning-based goal 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. Sohrabi *et al.* (2016) propose a probabilistic plan recognition approach that deals with unreliable observations (*i.e.*, noisy or missing observations), and recognizes both goals and plans. Unlike these last three approaches, which provide a probabilistic interpretation of the recognition problem, we do not deal with probabilities yet. Keren *et al.* (2014) develop an alternate view of the goal recognition problem, and rather than developing new goal recognition algorithms, they develop an approach that changes the domain definition to facilitate the goal recognition process. Their work could potentially be used alongside our techniques. Most recently, Pereira and Meneguzzi (2016) describe a landmark-based plan recognition approach, which was not as accurate and scalable as the one shown in this paper and did not use the notion of landmark uniqueness to improve recognition.

8 Conclusions

We developed a novel goal recognition approach that employs planning landmarks to compute progress towards a set of candidate goals from observations. This paper provides two contributions. First, a goal recognition algorithm that can leverage any landmark extraction technique to efficiently infer hidden goals; and second, we show that our heuristics surpass the state-of-the-art across a variety of planning domains. Both heuristics yield high accuracy consistent with the state-of-the-art, while taking significantly less time than earlier approaches to recognize the hidden goal. Importantly, our approach scales much better for larger plan lengths while maintaining accuracy, unlike the current state-of-the-art in the field (Ramírez and Geffner 2009).

We intend to explore multiple avenues for future work. First, we intend to explore other landmark extraction algorithms to obtain additional information from planning domains, such as temporal landmarks (Karpas et al. 2015). Second, we aim to associate a probabilistic interpretation to the observed landmarks and compare these to recent work, such as (E.-Martín, R.-Moreno, and Smith 2015) and (Sohrabi, Riabov, and Udrea 2016).

References

Avrahami-Zilberbrand, D., and Kaminka, G. A. 2005. Fast and Complete Symbolic Plan Recognition. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005).*

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Journal of Artificial Intelligence Research (JAIR)* 69:165–204.

E.-Martín, Y.; R.-Moreno, M. D.; and Smith, D. E. 2015. A Fast Goal Recognition Technique Based on Interaction Estimates. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI 2015).*

Geib, C. W., and Goldman, R. P. 2009. A Probabilistic Plan Recognition Algorithm Based on Plan Tree Grammars. *Artificial Intelligence* 173(11):1101–1132.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated Planning - Theory and Practice.* Elsevier.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *Journal of Artificial Intelligence Research (JAIR)* 22(1):215–278.

Karpas, E.; Wang, D.; Williams, B. C.; and Haslum, P. 2015. Temporal Landmarks: What Must Happen, and When. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015).*

Keren, S.; Gal, A.; and Karpas, E. 2014. Goal Recognition Design. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS 2014).*

Mirsky, R.; Stern, R.; Gal, Y. K.; and Kalech, M. 2016. Sequential Plan Recognition. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016).*

Pattison, D., and Long, D. 2010. Domain Independent Goal Recognition. In *Proceedings of the Fifth European Starting AI Researcher Symposium (STAIRS).*

Pereira, R. F., and Meneguzzi, F. 2016. Landmark-Based Plan Recognition. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016).*

Ramírez, M., and Geffner, H. 2010. Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 2010).*

Ramírez, M., and Geffner, H. 2009. Plan Recognition as Planning. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009).*

Richter, S., and Westphal, M. 2010. The LAMA Planner: Guiding Cost-based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research (JAIR)* 39(1):127–177.

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan Recognition as Planning Revisited. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016).*

Sukthankar, G.; Goldman, R. P.; Geib, C.; Pynadath, D. V.; and Bui, H. H. 2014. *Plan, Activity, and Intent Recognition: Theory and Practice*. Elsevier.