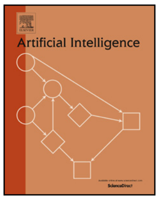
Artificial Intelligence 279 (2020) 103217

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint



Landmark-based approaches for goal recognition as planning

Ramon Fraga Pereira [a,*], Nir Oren [b], Felipe Meneguzzi [a]

[a] *School of Technology, Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil*
[b] *Department of Computing Science, University of Aberdeen, Aberdeen, Scotland, United Kingdom*

A R T I C L E   I N F O

Article history:
Received 1 January 2019
Received in revised form 21 November 2019
Accepted 28 November 2019
Available online 4 December 2019

Keywords:
Goal recognition
AI planning
Landmarks

A B S T R A C T

Recognizing goals and plans from complete or partial observations can be efficiently achieved through automated planning techniques. In many applications, it is important to recognize goals and plans not only accurately, but also quickly. To address this challenge, we develop novel goal recognition approaches based on planning techniques that rely on planning landmarks. In automated planning, landmarks are properties (or actions) that cannot be avoided to achieve a goal. We show the applicability of a number of planning techniques with an emphasis on landmarks for goal recognition tasks in two settings: (1) we use the concept of landmarks to develop goal recognition heuristics; and (2) we develop a landmark-based filtering method to refine existing planning-based goal and plan recognition approaches. These recognition approaches are empirically evaluated in experiments over several classical planning domains. We show that our goal recognition approaches yield not only accuracy comparable to (and often higher than) other state-of-the-art techniques, but also result in substantially faster recognition time over existing techniques.

1. Introduction

As computer systems become increasingly complex, coordination requires identifying and reasoning about the goals and plans of others. Goal and plan recognition can be seen as the task of recognizing goals and plans based on often partial observations that include actions executed by agents and properties of agent behavior in an environment [1]. Most goal and plan recognition approaches [2–5] employ plan libraries to represent agent behavior, i.e., a plan library with plans for achieving goals, resulting in approaches to recognize plans that are analogous to parsing. Such techniques have been used in applications including crime detection and prevention [6], monitoring activities in elder-care [7], recognizing plans in educational environments [8] and exploratory domains [9], and traffic monitoring [10], among others [6,11–13]. Existing work [14–22] use a planning domain definition (a domain theory) to represent potential agent behavior, bringing goal and plan recognition closer to planning algorithms. In doing so, these approaches allow techniques used in planning algorithms to be employed for recognizing goals and plans, thereby requiring less domain information.

The key limitation of the vast majority of the previous work in goal and plan recognition is that they require running costly planning algorithms, often multiple times, to recognize a goal. Our key contribution in this article is to describe a set of techniques that obviate running a full fledged planner to recognize goals, and, instead, use information from the structure of planning instances. Specifically, we develop recognition approaches that are based on automated planning techniques

* Corresponding author.
E-mail addresses: ramon.pereira@acad.pucrs.br (R.F. Pereira), n.oren@abdn.ac.uk (N. Oren), felipe.meneguzzi@pucrs.br (F. Meneguzzi).

https://doi.org/10.1016/j.artint.2019.103217
0004-3702/© 2019 Elsevier B.V. All rights reserved.

(without pre-defined static plan libraries) that rely on planning landmarks [23], namely, landmark-based approaches for goal recognition. In automated planning, landmarks are facts (or actions) that every plan must satisfy (or execute) at some point in every plan execution to achieve a goal. In this work, landmarks allow our recognition approaches to reason about what cannot be avoided for achieving goals, substantially speeding up recognition time. Our use of landmarks to drive goal recognition stems from properties of landmarks in classical planning, they are necessary conditions to achieving goals, and thus provide very strong evidence that certain observations are tied to specific goals. Although their computation is in theory expensive [23], in practice, we can efficiently compute very informative sets of ordered landmarks, and critically need to do so only once per goal recognition problem, resulting in a very efficient overall algorithm which avoids the use of automated planners, as most recognition approaches do.

In this article, we provide three additional contributions to the state-of-the-art in goal recognition. First, we develop two novel goal recognition heuristics that rely on landmarks and obviate the need to execute a planner yielding substantial run-time gains. Our initial heuristic estimates goal completion by considering the ratio between achieved and extracted landmarks of a candidate goal. We expand this heuristic to use a *landmark uniqueness value*, representing how common landmarks are among multiple goal hypotheses. Second, we develop a filtering method that rules out candidate goals by estimating how many landmarks required by every goal in the set of candidate goals have been reached within a sequence of observations. This filtering method can be applied to other planning-based goal and plan recognition approaches, such as the approaches from Ramírez and Geffner [14,15] (with a probabilistic ranking), as well as from Sohrabi et al. [19]. Third, we show the effectiveness of our technique both formally and empirically. Formally, we prove key properties of our recognition heuristics and their use as a filtering mechanism. We note that we build our goal recognition heuristics using the concept of *fact landmarks*, but such heuristics are not limited to fact landmarks for recognizing goals, they could be easily adapted to deal with *action landmarks*. Specifically, the landmark counting we use in our heuristics is agnostic to what kind of landmarks are being used, and the algorithms to extract fact landmarks necessarily also extract action landmarks.

We empirically evaluate our approaches using a set of well-known domains from the International Planning Competition (IPC), as well as a number of domains we developed specifically to measure the scalability of goal and plan recognition algorithms. For all domains and problems, we evaluate the approaches using datasets with varying degrees of observability (missing observations) and noise (spurious observations). We compare our heuristics approaches against the current state-of-the-art [14,15,18,19,24,25] by using a dataset developed by Ramírez and Geffner [14,15], and a new dataset we generated for other planning domains with larger and more complex problems, as well as problems with missing and noisy observations. Experiments show that our heuristic approaches 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 from a set of goal hypothesis is non-trivial.

Albrecht and Stone's survey [26] provide a classification of different goal and plan recognition approaches. Within this taxonomy, the approaches in this article fall into the category of "Plan Recognition by Planning in Domain Models". The authors note different properties for agents and environments for different recognition approaches. Given the similarity between our heuristics and that of Ramírez and Geffner [14,15], we permit stochastic actions, but assume unchanging behavior. Factors between agents are known and agents are assumed to be independent, with no need to assume common goals. There is no move ordering defined between agents, and our state/action representations are discrete, while state and action observations are assumed to be full.

The remainder of this article is organized as follows. Section 2 provides background on planning, domain-independent heuristics, landmarks, and goal recognition. We proceed to describe how we extract useful information from planning domain definitions in Section 3, which we use throughout the article. In Section 4, we develop our goal recognition approaches using landmarks. We empirically evaluate our approaches in Section 5, which shows the results of the experiments for our goal recognition approaches against the state-of-the-art. In Section 6, we survey related work and compare the state-of-the-art with our proposed approaches. Finally, in Section 7, we conclude this article by discussing limitations, advantages and future directions of our approaches.

2. Background

In this section, we review essential background on planning terminology and landmarks. Finally, we define the task of goal recognition over planning domain definitions.

2.1. Planning

Planning is the problem of finding a sequence of actions (i.e., a plan) that achieves a particular goal from an initial state. In this work, we adopt the terminology from Ghallab et al. [27] to represent planning domains and problems. First, we define a *state* in the environment as a set of ground predicates.

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 (τ_1, τ_2, ..., τ_n) – terms are either constants or variables. We refer to grounded predicates that represent logical values according to some interpretation as facts, which are divided into two types: positive and negated facts, as well as constants for truth ($\top$) and falsehood ($\bot$). A state S is a finite set of positive facts f that follows the closed world assumption so that if $f \in S$, then f is true in S. We assume a simple inference relation $\models$ such that $S \models f$ iff $f \in S$, $S \not\models f$ iff $f \notin S$, and $S \models f_1 \wedge ... \wedge f_n$ iff $\{f_1, ..., f_n\} \subseteq S$.

Planning domains describe the environment's dynamics through operators, which use a limited first-order logic representation to define schemata for state-modification actions.

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

We say an action a is applicable to a state S if and only if $S \models pre(a)$, and generates a new state S' such that $S' := (S \cup eff(a)^+ / eff(a)^-$.

Definition 3 *(Planning domain).* A planning domain definition Ξ is represented by a pair $\langle \Sigma, \mathcal{A}\rangle$, which specifies the knowledge of the domain model, and consists of a finite set of facts Σ (e.g., environment properties) and a finite set of actions $\mathcal{A}$.

A *planning instance*, comprises both a *planning domain* and the elements of a *planning problem*, describing a finite set of *objects* of the environment, the *initial state*, and the *goal state* which an agent wishes to achieve.

Definition 4 *(Planning instance).* A planning instance Π is represented by a triple $\langle \Xi, \mathcal{I}, G\rangle$.

- $\Xi = \langle \Sigma, \mathcal{A}\rangle$ is the domain definition;
- $\mathcal{I} \subseteq \Sigma$ 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 achieved.

Classical planning representations often separate the definition of $\mathcal{I}$ and G as part of a planning problem to be used together with a domain Ξ, such as STRIPS [28] and PDDL [29]. In this work, we use the STRIPS fragment of PDDL to formalize planning domains and problems. Finally, a *plan* is the solution of a planning instance.

Definition 5 *(Plan).* A plan π for a planning instance $\Pi = \langle \Xi, \mathcal{I}, G\rangle$ is a sequence of actions $\langle a_1, a_2, ..., a_n\rangle$ that modifies the initial state $\mathcal{I}$ into a state $S \models G$ in which the goal state G holds by the successive execution of actions in a plan π. A plan π^* with length $|\pi^*|$ is optimal if there exists no other plan π' for Π such that $\pi' < \pi^*$.

While actions can have an associated cost, we — as done in classical planning — assume that this cost is 1 for all instantiated actions. A plan π is considered optimal if its cost, and thus length, is minimal.

Finally, modern classical planners use a variety of heuristics to efficiently explore the search space of planning domains by estimating the cost to achieve a specific goal [27]. In classical planning, this estimate is often the number of actions to achieve the goal state from a particular state, so we describe all techniques assuming a uniform action cost $c(a) = 1$ for all $a \in \mathcal{A}$. Thus, the cost for a plan $\pi = [a_1, a_2, ..., a_n]$ is $c(\pi) = \Sigma c(a_i)$. Many heuristics used in planning are *admissible*, never overestimating the cost of achieving a goal. Formally, a heuristic $h(s)$ is admissible if and only if $h(s) \leq h^*(s)$ for all states s, where $h^*(s)$ is the optimal cost to achieve the goal when starting from state s. The use of an admissible heuristic guarantees that any plan found will be optimal for certain planning techniques. An inadmissible heuristic is one which is not admissible. Existing works use planning heuristics not only to build automated planners and explore search space but also for planning in adversarial and cooperative environments [30], plan optimality monitoring [31], and detecting commitment abandonment [32].

2.2. Landmarks

In the planning literature [33], *landmarks* are defined as necessary properties (alternatively, actions) that must be true (alternatively, executed) at some point in every valid plan (cf. Definition 5) to achieve a particular goal. Landmarks are often partially ordered according to the sequence in which they must be achieved. Hoffman et al. [23] define fact landmarks as follows.

Definition 6 *(Fact landmark).* Given a planning instance $\Pi = \langle \Xi, \mathcal{I}, G\rangle$, a formula[1] F_l is a landmark in Π iff F_l is true at some point along all valid plans that achieve G from $\mathcal{I}$. 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 (or achieved) at some point along all valid plan executions.

[1] In logic, a formula is a type of syntactic formula (conjunctive or disjunctive) with a sequence of symbols, which is well-formed and has a truth value.

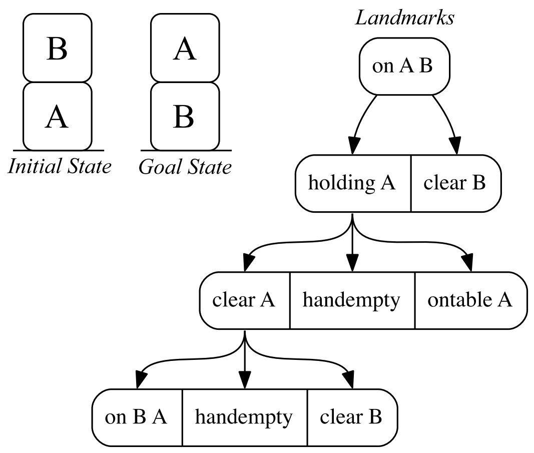


Fig. 1. Ordered landmarks for a Blocks-World problem instance.

Vidal and Geffner [34] define action landmarks as necessary actions that must be executed at some point along all valid plans that achieve a goal state G from an initial state $\mathcal{I}$. In this work, we do not explicitly use the concept of action landmarks, but rather use *fact landmarks* to build our goal recognition approaches.

From the concept of fact landmarks, Hoffmann et al. [23] introduce two types of landmarks as formulas: *conjunctive* and *disjunctive landmarks*. A *conjunctive landmark* is a set of facts that must be true together at some point in every valid plan to achieve a goal. A *disjunctive landmark* is a set of facts such that at least one of the facts must be true at some point in every valid plan to achieve a goal. Fig. 1 shows an example that illustrates a set of landmarks for a Blocks-World[2] problem instance. This example shows a set of conjunctive ordered landmarks (connected boxes in the figure) that must be true to achieve the goal state `(on A B)`. For instance, to achieve the fact landmark `(on A B)` which is also the goal state, the conjunctive landmark `(and (holding A) (clear B))` must be true immediately before the goal state.

Whereas in planning the concept of landmarks is used to build heuristics [33] and planning algorithms [35], in this work, we propose a novel use for landmarks: to build goal recognition approaches. Intuitively, we use landmarks as waypoints that agents must follow to achieve their goals.

2.3. Goal recognition

Goal recognition is the task of recognizing agents' goals by observing their interactions in an environment [1]. Such observed interactions (i.e., observations) comprise the evidence available to recognize goals. Definition 7 follows the formalism proposed by Ramírez and Geffner in [14,15], characterizing an observation sequence as the result of a sequence of actions.

Definition 7 *(Observation sequence).* An observation sequence $O = \langle o_1, o_2, ..., o_n \rangle$ is said to be satisfied by a plan $\pi = \langle a_1, a_2, ..., a_m \rangle$, if there is a monotonic function f that maps the observation indices $j = 1, ..., n$ into action indices $i = 1, ..., m$, such that $a_{f(j)} = o_j$.

By combining the various notions of planning problem and an observation sequences, we formally define a goal recognition problem over a planning domain definition following Ramírez and Geffner [14][3] in Definition 8, and provide a weak notion of solution to that problem in Definition 9.

Definition 8 *(Goal recognition problem).* A goal recognition problem is a tuple $T_{GR} = \langle \Xi, \mathcal{I}, \mathcal{G}, O \rangle$, where:

- $\Xi = \langle \Sigma, \mathcal{A} \rangle$ is a planning domain definition;
- $\mathcal{I}$ is the initial state;
- $\mathcal{G}$ is the set of possible goals, which is assumed to include a correct 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, such that O is satisfied by a valid plan π (from Definition 5), i.e., with each observation $o_i = name(a)$ for $a \in \mathcal{A}$, and the corresponding action a being part of π, and that π transitions $\mathcal{I}$ into G^* through the sequential execution of actions in π.

[2] Blocks-World is a classical planning domain where a set of stackable blocks must be re-assembled on a table [27].

[3] Unlike the probabilistic approach developed by Ramírez and Geffner [15], our heuristic approaches do not use any prior probabilities to perform the goal recognition process, we use a ranking with scores from 0 to 1 (see Section 4).

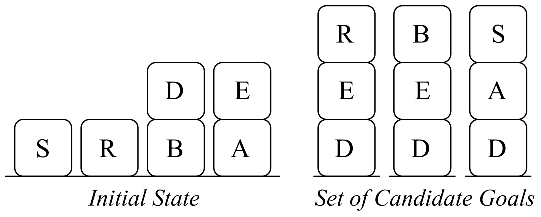


Fig. 2. BLOCKS-WORLD example.

Definition 9 *(Solution to a goal recognition problem).* A solution to a goal recognition problem $T_{GR} = \langle \Xi, \mathcal{I}, \mathcal{G}, O \rangle$ is a nonempty subset of the set of possible goals $\mathbf{G} \subseteq \mathcal{G}$ such that $\forall G \in \mathbf{G}$ there exists a plan π_G generated from a planning instance $\langle \Xi, \mathcal{I}, G \rangle$ and O is satisfied by π_G.

Clearly, the most desirable solution for a goal recognition problem is a singleton set containing only the correct hidden goal $G^* \in \mathcal{G}$ that the observation sequence O of a plan execution achieves. A solution is less than ideal in this context if it contains more than one goal, as all such goals are equally likely. Indeed, we measure this in the experiments (Section 5) as the *Spread* in $\mathcal{G}$. As an example of how the goal recognition process works, consider the following example.

Example 1. To exemplify the goal recognition process, let us consider the BLOCKS-WORLD example illustrated in Fig. 2. The initial state represents an initial configuration of stackable blocks, while the set of candidate goals is made up of different possible stacks of blocks spelling out the following words: `RED`, `BED`, *and* `SAD`. Consider an observation sequence for a hidden goal resulting in the word `RED` consisting of the following action sequence: `[(unstack D B), (putdown D), (unstack E A), (stack E D), (pickup R), (stack R E)]`. By following the full plan, we can easily infer that the hidden goal is indeed `RED`. However, if we cannot observe action `(stack R E)`, it is not trivial to infer that `RED` is indeed the goal that the observed sequence aims to achieve. Instead, the observations suggest that more than one goal is being pursued, and this occurs because the action `(stack R E)` has some landmarks that are important to disambiguate and infer that `RED` is in fact the correct goal.

Like most planning-based recognition approaches [14,15,18,19], we also deal with missing observations during the goal recognition process. However, we differ from [19] in that we do not deal with noisy (unreliable) observations explicitly. Nevertheless our technique proves to be robust against noise by relying exclusively on necessary conditions for the plans leading to each goal as our empirical analysis in Section 5 corroborates. In a partial observation sequence, we observe only a potentially disjoint sub-sequence of actions of a plan that achieves a particular goal because some actions are missing or obfuscated. By contrast, a noisy observation sequence contains one or more actions (or a set of facts) that might not be part of a plan that achieves a particular goal, e.g., when a sensor fails and generates abnormal or spurious readings. We formalize the way in which an environment generates observations of agent plans in Definition 10.

Definition 10 *(Observation sequence generation).* Let $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ be a plan generated by a planning instance $\Pi = \langle \Xi, \mathcal{I}, G \rangle$. An action projection function $ap(a) : \mathcal{A} \mapsto \vec{\mathcal{A}}$ is a function that maps actions to sequences of zero or more actions. An observation sequence generation function $os(\pi)$ is a function mapping a plan π into an observation sequence:

$$os(\pi) = \begin{cases} \langle\rangle & \text{if } \pi = \langle\rangle \\ \langle ap(a_1) \rangle \cdot os(\langle a_2, \ldots, a_n \rangle) & \text{if } \pi = \langle a_1, \ldots, a_n \rangle \end{cases}$$

The key to generating such sequences is how the rules for function ap translate actions. Following our Example 1, we could define an action projection function that never generates observations for unstack actions, and generates noise for all stack actions as follows.

$$ap_1(a) = \begin{cases} \langle \texttt{(pickup X)} \rangle & \text{if } a = \texttt{(pickup X)} \\ \langle \texttt{(putdown X)} \rangle & \text{if } a = \texttt{(putdown X)} \\ \langle\rangle & \text{if } a = \texttt{(unstack X Y)} \\ \langle \texttt{(stack X Y)}, \texttt{(unstack X Y)} \rangle & \text{if } a = \texttt{(stack X Y)} \end{cases}$$

We formally define missing and noisy observations in Definitions 11 and 12.

Definition 11 *(Missing observation).* Let $\Pi = \langle \Xi, \mathcal{I}, G \rangle$ be a planning instance, π be a valid plan that achieves G from $\mathcal{I}$, and O is an observation sequence induced by an observation generation function os with an action projection function ap. An observation sequence O misses observations (is a partial or incomplete observation sequence) with respect to the plan π

that achieves the goal G from $\mathcal{I}$ if the ap function contains a mapping $a \mapsto \langle\rangle$ for some action a, i.e., it maps one or more actions into the empty sequence.

Definition 12 *(Noisy observation).* Let $\Pi = \langle \Xi, \mathcal{I}, G\rangle$ be a planning instance, π be a valid plan that achieves G from $\mathcal{I}$, and O is an observation sequence induced by an observation generation function os with an action projection function ap. An observation sequence O contains noisy observations with respect to the plan π that achieve the goal G from $\mathcal{I}$ if the ap function maps any action a into a non-empty sequence containing one or more actions $a_i \neq a$. An observation is said to be **noiseless** if function $ap(a)$ is such that either for all actions $a \in \mathcal{A}$, $ap(a) = \langle a\rangle$ or $ap(a) = \langle\rangle$.

We say that an observation sequence with no missing observations is *full*. It should be noted that there is no information in the observation sequence O about which actions are missing or noise. Thus, the recognizer receives no information about how complete an observation sequence is. An observation sequence with noisy observations therefore simply contains additional (spurious) actions within it.

Example 2. Assume that a valid plan to achieve a goal G is $\pi = [a, b, c, d, e]$. Consider the following observation sequences O_{m1}, O_{m2}, and O_{m3}:

- $O_{m1} = [a, d]$;
- $O_{m2} = [b, e]$; and
- $O_{m3} = [d, a, c]$.

Observation sequences O_{m1} and O_{m2} satisfy Definition 11, and therefore, they are partial observation sequences and contain missing observed actions. O_{m3} is not a partial observation sequence because it does not satisfy Definition 11 as the observation sequence $[d, a, c]$ is not a strict subset of ordered actions of the plan π.

Now, consider the following observation sequences O_{n1} and O_{n2}:

- $O_{n1} = [a, b, c, d, e, g]$; and
- $O_{n2} = [b, d, h]$.

Both observation sequences O_{n1} and O_{n2} contain noisy observations (g and h respectively) and satisfy Definition 11. However, note that O_{n2} contains not only noisy observations but it also misses observations, i.e., O_{n2} is a partial observation sequence with noisy observations.

Although we define missing and noisy observations with actions as observations, our goal recognition approaches can also deal with facts (or fluents) as observations, like [19]. Indeed, as we see in Section 4, using states as observations makes goal recognition much easier for our heuristic approaches, since we can use the observations directly to compute achieved landmarks. In Section 4, we show that what matters for our goal recognition approaches is the evidence of fact landmarks during the observations, and it is irrelevant whether this evidence is provided by either an observed action or a set of facts.

3. Extracting recognition information from planning definition

In this section, we describe the process through which we extract useful information for goal recognition from a planning domain definition. In Section 3.1 we describe landmark extraction algorithms from the literature, and how we use these algorithms in our approaches. Then, in Section 3.2, we describe how we classify facts into partitions from planning action descriptions and how we use them during the goal recognition process.

3.1. Extracting landmarks

Hoffman et al. [23] proves that the process of extracting exactly all landmarks and deciding about their ordering is PSPACE-complete, which is equivalent to the complexity of deciding plan existence [36]. Nevertheless, most extraction algorithms extract only a subset of landmarks for a given planning instance for efficiency. While there are several algorithms to extract landmarks and their orderings in the literature that we could use [37,33,38], we chose the extraction algorithm from Hoffmann et al. [23] to extract landmarks from planning instances due to its speed and simplicity. This algorithm can extract both conjunctive and disjunctive landmarks, but we use the conjunctive landmarks to build heuristics for goal recognition.

To represent landmarks and their ordering, the algorithm of Hoffmann et al. [23] 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. This algorithm uses a Relaxed Planning Graph (RPG) [39], which is a leveled graph that ignores the delete-list effects of all actions, thus containing no mutex relations. Once the RPG is built, the algorithm extracts *landmark candidates*

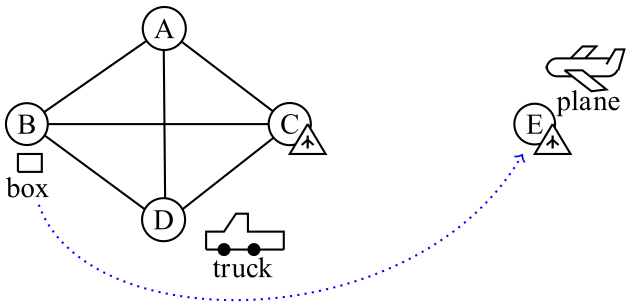


Fig. 3. LOGISTICS problem example with goal `(at BOX AIRPORT-E)`.

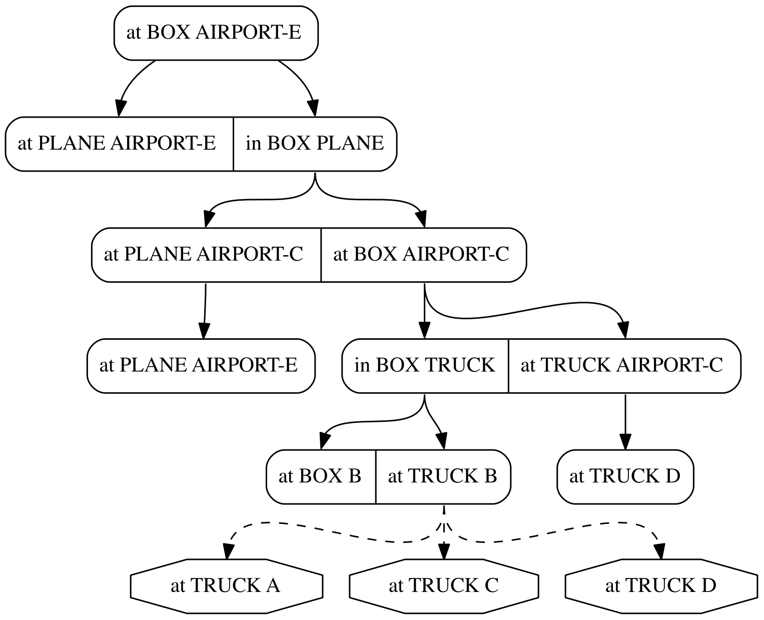


Fig. 4. Ordered fact landmarks extracted from LOGISTICS problem example shown in Fig. 3. Fact landmarks that must be true together are represented by connected boxes, which are conjunctive facts, i.e., representing conjunctive landmarks. Disjunctive landmarks are represented by octagon boxes that are connected by dashed lines.

by backwards-chaining from the RPG level in which all facts of the goal state G are possible, and, for each fact g in G, checks which facts must be true until the first level of the RPG. For example, if fact B is a landmark and all actions that achieve B share A as precondition, then A is a landmark candidate. To confirm that a landmark candidate is indeed a landmark, the algorithm builds a new RPG structure by removing actions that achieve this landmark candidate and checks the solvability over this modified problem,[4] and, if the modified problem is unsolvable, then the landmark candidate is a necessary landmark. This means that the actions that achieve the landmark candidate are necessary to solve the original planning problem.

To exemplify the output of the landmark extraction algorithm from [23], consider the LOGISTICS[5] problem example shown in Fig. 3. Fact landmarks extracted for this example are shown respectively in Listing 1 and Fig. 4. While Listing 1 shows one possible serialization of the landmarks, Fig. 4 represents the same landmarks ordered from bottom-up by facts that must be true together. These landmarks allows us to track way-points during a plan execution to determine which goals this plan is going to achieve.

This landmark extraction algorithm is referred to as 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 $\mathcal{G}$ or a single goal G. In case the input is a set of candidate goals $\mathcal{G}$, 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).

[4] Deciding the solvability of a relaxed planning problem using an RPG structure can be done in polynomial time [40].
[5] The LOGISTICS domain consists of airplanes and trucks transporting packages between locations (e.g., airports and cities).

```
Fact Landmarks:
(and (at BOX AIRPORT–E))
(and (at PLANE AIRPORT–E) (in BOX PLANE))
(and (at PLANE AIRPORT–C) (at BOX AIRPORT–C))
(and (at PLANE AIRPORT–E))
(and (at TRUCK D))
(and (in BOX TRUCK) (at TRUCK AIRPORT–C))
(and (at BOX B) (at TRUCK B))
(or (at TRUCK A) (at TRUCK C) (at TRUCK D))
```

Listing 1: Fact landmarks (conjunctive and disjunctive) extracted from the Logistics example in Fig. 3.

We note that many landmark extraction techniques, including that of Hoffmann et al. [23], might infer incorrect landmark orderings, which can lead to problems if the goal recognition process relies on the ordering information to make inferences. Nevertheless, even using the possibly "incorrect" landmark orderings extracted from the algorithm of Hoffmann et al. [23], our empirical evaluation shows that the orderings do not affect detection performance in the experimental datasets. We discuss landmark orderings further in Section 4.1.

3.2. Classifying facts into partitions

Pattison and Long [16] classify facts into mutually exclusive partitions in order to infer whether certain observations are likely to be goals for goal recognition. Their classification relies on the fact that, in some planning domains, predicates may provide additional information that can be extracted by analyzing preconditions and effects in operator definitions. We use this classification to infer if certain observations are consistent with a particular goal, and if not, we can eliminate a candidate goal. We formally define fact partitions in what follows.

Definition 13 *(Strictly activating).* A fact f is strictly activating if $f \in \mathcal{I}$ and $\forall a \in \mathcal{A}$, $f \notin \mathit{eff}(a)^+ \cup \mathit{eff}(a)^-$. Furthermore, $\exists a \in \mathcal{A}$, such that $f \in \mathit{pre}(a)$.

Definition 14 *(Unstable activating).* A fact f is unstable activating if $f \in \mathcal{I}$ and $\forall a \in \mathcal{A}$, $f \notin \mathit{eff}(a)^+$ and $\exists a, b \in \mathcal{A}$, $f \in \mathit{pre}(a)$ and $f \in \mathit{eff}(b)^-$.

Definition 15 *(Strictly terminal).* A fact f is strictly terminal if $\exists a \in \mathcal{A}$, such that $f \in \mathit{eff}(a)^+$ and $\forall a \in \mathcal{A}$, $f \notin \mathit{pre}(a)$ and $f \notin \mathit{eff}(a)^-$.

A *Strictly Activating* fact (Definition 13) appears as a precondition, and does not appear as an add or delete effect in an operator definition. This means that unless defined in the initial state, this fact can never be added or deleted by an operator. An *Unstable Activating* fact (Definition 14) appears as both a precondition and a delete effect in two operator definitions, so once deleted, this fact cannot be re-achieved. The deletion of an unstable activating fact may prevent a plan execution from achieving a goal. A *Strictly Terminal* fact (Definition 15) does not appear as a precondition of any operator definition, and once added, cannot be deleted. For some planning domains, this kind of fact is the most likely to be in the set of goal facts, because once added in the current state, it cannot be deleted, and remains true until the final state.

The fact partitions that we can extract depend on the planning domain definition. For example, from the Blocks-World domain, it is not possible to extract any fact partitions. However, it is possible to extract fact partitions from the IPC-Grid[6] domain, such as *Strictly Activating* and *Unstable Activating* facts. In this work, we use fact partitions to obtain additional information on fact landmarks during the goal recognition process. For example, consider an *Unstable Activating* fact landmark L_{ua}. 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 after deletion, because there is no available action that achieves L_{ua} again.

4. Landmark-based goal recognition

We now describe goal recognition approaches that rely on planning landmarks. We start with a method to track and compute the evidence of landmarks from observations in Section 4.1. After that, we describe how we build our goal recognition heuristics using planning landmarks in Sections 4.2 and 4.3. Finally, we develop a landmark-based filtering method that can be used with any other planning-based goal and plan recognition approach in Section 4.4.

4.1. Computing achieved landmarks in observations

An essential part of our approaches to goal recognition is the ability to track and compute the evidence of achieved fact landmarks in the observations. To do so, we compute the evidence of achieved fact landmarks in preconditions and effects

[6] IPC-Grid domain consists of an agent that moves in a grid using keys to open locked locations.

of observed actions during a plan execution [21] using the ComputeAchievedLandmarks function shown in Algorithm 1. This algorithm 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 (provided by the ExtractLandmarks function which computes fact landmarks given a planning domain). Note that Algorithm 1 can be easily modified to allow it to deal with observations as states, so instead of analyzing preconditions and effects of actions, we compare the observations directly to computed landmarks.

Algorithm 1 Compute Achieved Landmarks in 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:     $\Lambda_{\mathcal{G}} := \langle\rangle$
3:     for each goal $G$ in $\mathcal{G}$ do                ▷ Map goals $\mathcal{G}$ to their respective achieved landmarks.
4:         $\mathcal{L}_G :=$ fact landmarks of $G$ s.t $\langle G, \mathcal{L}_G\rangle$ in $\mathcal{L}_{\mathcal{G}}$
5:         $\mathcal{L}_{\mathcal{I}} :=$ all fact landmarks $L \in \mathcal{I}$
6:         $\mathcal{L} := \emptyset$
7:         for each observed action $o$ in $O$ do
8:             $\mathcal{L} := \{L \in \mathcal{L}_G | L \in pre(o) \cup pre(o)^+ \wedge L \notin \mathcal{L}\}$
9:             $\mathcal{L}_{\prec} :=$ predecessors $L_{\prec}$ of all $L \in \mathcal{L}$, s.t $L_{\prec} \notin \mathcal{L}$
10:            $\mathcal{AL}_G := \mathcal{AL}_G \cup \{\mathcal{L}_{\mathcal{I}} \cup \mathcal{L} \cup \mathcal{L}_{\prec}\}$
11:        $\Lambda_{\mathcal{G}}(G) := \mathcal{AL}_G$                ▷ Achieved landmarks of $G$.
12:    return $\Lambda_{\mathcal{G}}$
```

Algorithm 1 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 7 to 10. For each observed action o in O, the algorithm computes all fact landmarks of G that are either in the preconditions or effects of o in Line 8. 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 9. 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)`). 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 10. 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 12). Example 3 illustrates the execution of Algorithm 1 to compute achieved landmarks from the observations of our running example.

Example 3. Consider the Blocks-World example from Fig. 2, and the following observed actions: `(unstack E A)` and `(stack E D)`. From these observed actions, the candidate goal `RED`, and the set of fact landmarks of this candidate goal (Fig. 5), 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. Fig. 5 shows the set of achieved landmarks for the word `RED` in gray. Listing 3 shows in bold the set of achieved landmarks that our algorithm computes for the set of candidate goals in Fig. 2.

The complexity of computing achieved landmarks in observations (Algorithm 1) with the process of extracting landmarks (EL) is: $O(EL + |\mathcal{G}| \cdot |O| \cdot |\mathcal{L}_{\mathcal{G}}|)$, where $\mathcal{G}$ is the set of candidate goals, O is the observation sequence, and $\mathcal{L}_{\mathcal{G}}$ is the extracted landmarks for $\mathcal{G}$. The complexity of our approach is dominated by the complexity of EL, and thus, given a suitable implementation of EL, our approach has polynomial complexity.

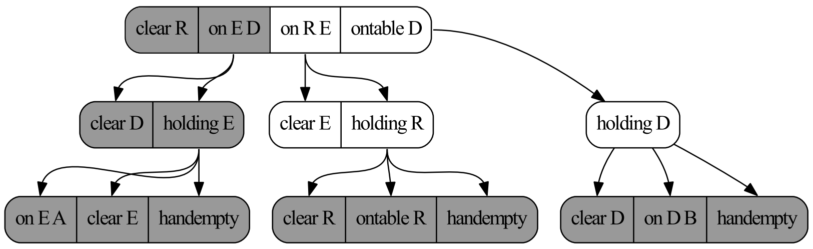


Fig. 5. Ordered fact landmarks extracted for the word RED. Fact landmarks that must be true together are represented by connected boxes. Connected boxes in grey represent achieved fact landmarks. Edges represent prerequisites between landmarks.

4.2. Landmark-based goal completion heuristic

We now describe a recognition heuristic that estimates the percentage of completion of a goal based on the number of landmarks that have been detected, and are required to achieve that goal [21]. This estimate represents the percentage of sub-goals in a goal that have been accomplished based on the evidence of achieved fact landmarks in the observations. We note that a candidate goal is composed of sub-goals comprised of the atomic facts that are part of a conjunction of facts in the goal definition.

Our heuristic method estimates the percentage of completion towards a goal by using the set of achieved landmarks computed by Algorithm 1 (ComputeAchievedLandmarks). Namely, this heuristic operates by aggregating the percentage of completion of each sub-goal into an overall percentage of completion for all facts of a goal. We denote this heuristic as h_{gc}, and it is formally defined by Equation (1), where $\mathcal{AL}_g$ is the number of achieved landmarks from observations of every sub-goal g of a 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 $\mathcal{L}_G$:

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

Thus, heuristic h_{gc} 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 T_{GR}, as well as a threshold value θ. The θ threshold gives us flexibility to avoid eliminating candidate goals whose percentage of goal completion are close to the highest completion value. In Line 2, the algorithm uses the ExtractLandmarks function to extract fact landmarks for all candidate goals. By taking as input the initial state $\mathcal{I}$, the observations O, and the extracted landmarks $\mathcal{L}_\mathcal{G}$, in Line 3, our algorithm first 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. Example 4 shows how heuristic h_{gc} estimates the completion of a candidate goal.

Algorithm 2 Recognize goals using the Goal Completion 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:     $\mathcal{L}_\mathcal{G} :=$ ExtractLandmarks($\Xi, \mathcal{I}, \mathcal{G}$)
3:     $\Lambda_\mathcal{G} :=$ ComputeAchievedLandmarks($\mathcal{I}, \mathcal{G}, O, \mathcal{L}_\mathcal{G}$)
4:     $maxh := \max_{G' \in \mathcal{G}} h_{gc}(G', \Lambda_\mathcal{G}(G'), \mathcal{L}_\mathcal{G}(G'))$
5:     return all $G$ s.t $G \in \mathcal{G}$ and
           $h_{gc}(G, \Lambda_\mathcal{G}(G), \mathcal{L}_\mathcal{G}(G)) \geq (maxh - \theta)$
```

Example 4. As an example of how heuristic h_{gc} estimates goal completion of a candidate goal, recall the Blocks-World example from Fig. 2. Consider that among these candidate goals (RED, BED, and SAD) the correct hidden goal is RED, and we observe the following partial sequence of actions: `(unstack E A)` and `(stack E D)`. Thus, based on the achieved landmarks $\mathcal{AL}_{\text{RED}}$ computed using Algorithm 1 (Fig. 5), our heuristic h_{gc} estimates that the percentage of completion for the goal RED is 0.66: `(clear R)` $= \frac{1}{1} +$ `(on E D)` $= \frac{3}{3} +$ `(on R E)` $= \frac{1}{3} +$ `(ontable D)` $= \frac{1}{3}$, and hence, $\frac{2.66}{4} = 0.66$. For the words BED and SAD our heuristic h_{gc} estimates respectively, 0.54 and 0.58.

Besides extracting landmarks for every candidate goal (EL), our landmark-based goal completion approach iterates over the set of candidate goals $\mathcal{G}$, the observations sequence O, and the extracted landmarks $\mathcal{L}_{\mathcal{G}}$. The heuristic computation of h_{gc} (HC) is linear on the number of fact landmarks. Thus, the complexity of this approach is: $O(EL + |\mathcal{G}| \cdot |O| \cdot |\mathcal{L}_{\mathcal{G}}| + HC)$. Finally, the goal ranking based on h_{gc} always ensures (under full observability) that the correct goal ranks highest (i.e., it is sound), with possible ties, as stated in Theorem 1.

Theorem 1 *(Soundness of the h_{gc} goal recognition heuristic). Let $T_{GR} = \langle \Xi, \mathcal{I}, \mathcal{G}, O \rangle$ be a goal recognition problem with candidate goals $\mathcal{G}$ such that $\forall G_1, G_2 \in \mathcal{G}, G_1 \neq G_2 \rightarrow G_1 \not\subset G_2$, a complete and noiseless observation sequence $O = \langle o_1, o_2, ..., o_n \rangle$. If $G^* \in \mathcal{G}$ is the correct hidden goal, then, for any landmark extraction algorithm that generates fact landmarks $\mathcal{L}_{\mathcal{G}}$ and computed landmarks $\Lambda_{\mathcal{G}}$, the estimated value of h_{gc} will always be highest for the correct hidden goal G^*, i.e., $\forall G \in \mathcal{G}$ it is the case that $h_{gc}(G^*, \Lambda_{\mathcal{G}}(G^*), \mathcal{L}_{\mathcal{G}}(G^*)) \geq h_{gc}(G, \Lambda_{\mathcal{G}}(G), \mathcal{L}_{\mathcal{G}}(G))$.*

Proof. The proof is straightforward from the definition of fact landmarks ensuring they are necessary conditions to achieve a goal G and that all facts $g \in G$ are necessary. Let us first assume that any pair of goals $G_1, G_2 \in \mathcal{G}$ are different, i.e., $G_1 \cap G_2 \neq \emptyset$, and that no action a in the domain Ξ achieves facts that are in any pair of goals simultaneously. Since any landmark extraction algorithm includes all facts $g \in G_1$ as landmarks for a goal G_1, then, for every other goal G_2, there exists at least one fact g such that $g \in G_1 \wedge g \notin G_2$ that sets it apart from G_2. Under these circumstances, an observation sequence O for the correct goal G^* will have achieved a set of landmarks $\Lambda_{\mathcal{G}}(G^*)$ that is exactly the same as the complete computed set of landmarks $\mathcal{L}_{\mathcal{G}}(G^*)$ for G^*. Hence $h_{gc}(G^*, \Lambda_{\mathcal{G}}(G^*), \mathcal{L}_{\mathcal{G}}(G^*)) = 1$, and $h_{gc}(G, \Lambda_{\mathcal{G}}(G), \mathcal{L}_{\mathcal{G}}(G)) < 1$ for any other goal $G \in \mathcal{G}$, since the numerator of the h_{gc} computation will be missing fact g for G as g is not a landmark of G. If we drop the assumption about the actions not achieving facts simultaneously in any pair of goals or that goals are identical, it is possible that $h_{gc}(G^*, \Lambda_{\mathcal{G}}(G^*), \mathcal{L}_{\mathcal{G}}(G^*)) = h_{gc}(G, \Lambda_{\mathcal{G}}(G), \mathcal{L}_{\mathcal{G}}(G)) = 1$, which still ensures that the under h_{gc}, G^* always ranks at the top, possibly tied with other goals. □

Thus, our goal completion heuristic is sound under full observability in the sense that it can never rank the wrong goal higher than the correct goal when we observe the landmarks. We note that there is one specific case when our landmark approach can provide wrong rankings, but which we explicitly exclude from the theorem, which is when the set of candidate goals contains two goals such that one is a sub-goal of the other (i.e., $G_1, G_2 \in \mathcal{G}$ and $G_1 \subseteq G_2$). In this case, any kind of "distance" to goal metric will report G_1 as being more likely than G_2 until the observations take the observed agent to G_2 (and these two goals will be tied in the heuristic). We close this section by commenting on the effect of landmark orderings on the accuracy of the heuristic. Specifically, although we do use the landmark order to infer the achievement of necessary prior landmarks that were not observed under missing observations, our heuristic does not consider the actual ordering of the landmarks. We infer prior landmarks to obtain more landmarks when we deal with partial observability. Nevertheless, we have experimented with different scoring mechanisms to account for landmarks having — or not having — been observed in the expected order, and these showed almost no advantage over the current heuristic. Consequently, although there are various different algorithms that generate better landmark orderings [23], the way in which we use the landmarks does not seem to be affected by more or less accurate landmark orderings.

There are two additional properties provable for our h_{gc} heuristic, first, given how our heuristic accounts for landmarks, the value outputted by the heuristic is strictly increasing as observations increase in length.

Proposition 1 *(Monotonicity of h_{gc}). The value of h_{gc} is monotonically (non-strictly) increasing in the observation sequence.*

Proof. By definition, $\mathcal{AL}_G$ is monotonically increasing, while all other values in h_{gc} remain constant. Therefore, from Equation (1), it is clear that h_{gc} must increase. □

Further, a corollary of Theorem 1 is that, under full observation, only the correct goal can reach a heuristic value of 1. This also illustrates why we restrict the theorem to settings where candidate goals are not subgoals of each other. Consider a goal to be at position d, and another to be at position g, with landmarks a, b, c, d, e, f, g. Since d itself is a landmark of g, d is implicitly a subgoal of g. If we observe all landmarks in an observation, then $h_{gc}(d) = \frac{4}{4} = 1$, and $h_{gc}(g) = \frac{7}{7} = 1$, which leads to Corollary 1.

Corollary 1. *If the goal being recognized has no subgoals being recognized under full observability, then $h_{gc} = 1$ iff the goal the heuristic is recognizing has been achieved.*

Proof. $h_{gc} = 1$ when $\frac{\sum_g \frac{|\mathcal{AL}_G|}{\mathcal{L}}}{|G|} = 1$, which can only occur when $|\mathcal{AL}_g| = |\mathcal{L}_g|$ for all $g \in G$. This clearly occurs when the goal being recognized is achieved. However, if the heuristic is also recognizing a subgoal, then this condition can be satisfied for the subgoal, hence the exception in the proposition. □

4.3. Landmark-based uniqueness heuristic

We now turn our attention to another heuristic which uses a measure of the uniqueness of landmarks. Many goal recognition problems contain multiple candidate goals that share common fact landmarks, generating ambiguity for our previous approaches. Clearly, landmarks that are common to multiple candidate 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. To develop this heuristic based on this intuition, 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 [21], and lies in the range (0,1]. For example, consider a landmark L that occurs only for a single goal within a set of candidate goals; since such a landmark is clearly unique, its uniqueness value is maximal (i.e., 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 $\mathcal{L_G}$.

Using the *landmark 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. This algorithm effectively 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 calculate the uniqueness value for every extracted landmark in the set of landmarks of the candidate goals using Equation (2). This computes the landmark uniqueness value of every landmark L of $\mathcal{L_G}$ and store it into Υ_{uv}. This heuristic is denoted as h_{uniq} and formally defined in Equation (3).

$$L_{Uniq}(L, \mathcal{L_G}) = \left(\frac{1}{\sum\limits_{\mathcal{L} \in \mathcal{L_G}} |\{L | L \in \mathcal{L}\}|} \right), \tag{2}$$

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

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 $\mathcal{L_G}$ (Line 2), and computes the set of achieved landmarks based on the observations, storing these in $\Lambda_{\mathcal{G}}$. Unlike Algorithm 2, in Line 6 this algorithm computes the landmark uniqueness value for every landmark L in $\mathcal{L_G}$ 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. Example 5 shows how heuristic h_{uniq} uses the concept of landmark uniqueness value to goal recognition.

Algorithm 3 Recognize goals using Uniqueness 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:     $\mathcal{L_G} :=$ EXTRACTLANDMARKS($\Xi, \mathcal{I}, \mathcal{G}$)
3:     $\Lambda_{\mathcal{G}} :=$ COMPUTEACHIEVEDLANDMARKS($\mathcal{I}, \mathcal{G}, O, \mathcal{L_G}$)
4:     $\Upsilon_{uv} := \langle\rangle$                              ▷ Map of landmarks to their uniqueness value.
5:     for each fact landmark $L$ in $\mathcal{L_G}$ do
6:         $\Upsilon_{uv}(L) := L_{Uniq}(L, \mathcal{L_G})$
7:     $maxh := \max_{G' \in \mathcal{G}} h_{uniq}(G', \Lambda_{\mathcal{G}}(G'), \mathcal{L_G}(G'), \Upsilon_{uv})$
8:     return all $G$ s.t $G \in \mathcal{G}$ and
           $h_{uniq}(G, \Lambda_{\mathcal{G}}(G), \mathcal{L_G}(G), \Upsilon_{uv}) \geq (maxh - \theta)$
```

Example 5. Recall the BLOCKS-WORLD example from Fig. 2 and consider the following observed actions: `(unstack E A)` and `(stack E D)`. Listing 2 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 2), our heuristic h_{uniq} estimates the following percentage for each candidate goal: $h_{uniq}(\texttt{RED}) = \frac{3.66}{6.33} = 0.58$; $h_{uniq}(\texttt{BED}) = \frac{2.66}{6.33} = 0.42$; and $h_{uniq}(\texttt{SAD}) = \frac{3.66}{8.33} = 0.44$. 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) (ontable B) (handempty)] = 1.0, [(on B E)] = 1.0,
[(clear B)] = 1.0, [(clear E) (holding B)] = 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 2: Extracted fact landmarks for the Blocks-World example in Fig. 2 and their respective uniqueness value.

Similar to our landmark-based goal completion approach, this approach iterates over the set of candidate goals $\mathcal{G}$, the observations sequence O, and the extracted landmarks $\mathcal{L}_{\mathcal{G}}$. However, in this approach we weight each landmark by how common this landmark is across all goal hypotheses. We call this weight the uniqueness value ($CLUniq$) and its computation is linear on the number of landmarks. The heuristic computation of h_{uniq} (HC) is also linear on the number of fact landmarks. Thus, the complexity of this approach is: $O(EL + |\mathcal{G}| \cdot |O| \cdot |\mathcal{L}_{\mathcal{G}}| + CLUniq + HC)$. Finally, since this is just a weighted version of the h_{gc} heuristic, it follows trivially from Theorem 1 that, for full observations, h_{uniq} always ranks the correct goal G^* highest.

Corollary 2 *(Correctness of h_{uniq} goal recognition heuristic). Let $T_{GR} = \langle \Xi, \mathcal{I}, \mathcal{G}, O\rangle$ be a goal recognition problem with candidate goals $G \in \mathcal{G}$, a complete and noiseless observation sequence $O = \langle o_1, o_2, ..., o_n\rangle$. If $G^* \in \mathcal{G}$ is the correct goal, then, for any landmark extraction algorithm that generates fact landmarks $\mathcal{L}_{\mathcal{G}}$ and computed landmarks $\Lambda_{\mathcal{G}}$, the estimated value of h_{uniq} will always be highest for the correct goal G^*, more specifically, $\forall G \in \mathcal{G}$ it is the case that $h_{uniq}(G^*, \Lambda_{\mathcal{G}}(G^*), \mathcal{L}_{\mathcal{G}}(G^*)) \geq h_{uniq}(G, \Lambda_{\mathcal{G}}(G), \mathcal{L}_{\mathcal{G}}(G))$.*

4.4. Filtering candidate goals from achieved landmarks in observations

We now develop an approach 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 [20]. This filtering method 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 method is detailed in function FilterCandidateGoals of Algorithm 4. This algorithm takes as input a goal recognition problem T_{GR}, which is composed of a planning domain definition Ξ, an initial state $\mathcal{I}$, a set of candidate goals $\mathcal{G}$, a set of observed actions O, and a filtering threshold θ. Our algorithm iterates over the set of candidate goals $\mathcal{G}$, and, for each goal G in $\mathcal{G}$, it extracts and classifies fact landmarks and partitions for G from the initial state $\mathcal{I}$ (Lines 4 and 5). Function PartitionFacts($\mathcal{L}_g, \mathcal{A}$) takes a set of goals and the actions in the domain and returns the fact partitions induced by the actions in $\mathcal{A}$ into the sets F_{sa} of strictly activating (from Definition 13), F_{ua} of unstable activating (from Definition 14), and F_{st} of strictly terminal (from Definition 15) facts. 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 $\mathcal{I}$, then the candidate goal G is no longer achievable, so we can 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 observed actions in O, we move on to checking landmarks of G within the observed actions in O. If we observe any landmarks in the preconditions and positive effects of the observed actions (Line 16), we compute the evidence of achieved landmarks for the candidate goal G (Line 18). Like Algorithm 1, we deal with missing observations by inferring that the unobserved predecessors of observed fact landmarks must have been achieved in Line 17. Given the number of achieved fact landmarks of G, we then 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 computing the percentage of landmark completion for all candidate goals in $\mathcal{G}$, we return the goals with the highest percentage of achieved landmarks within our filtering threshold θ (Line 22). We follow Definition 6 of fact landmarks and consider conjunctive landmarks as a single landmark when counting achieved landmarks (Line 21), except for the sub-goals, where each fact is a separate landmark. With respect to the threshold value, note that, if threshold $\theta = 0$, the filter returns only the goals with maximum completion, given the observations. The threshold gives us flexibility when dealing with missing observations and sub-optimal plans, which, when $\theta = 0$, may cause some potential candidate goals to be filtered out before we get additional observations. Example 6 shows how our filtering method efficiently prunes goals from a set of candidate goals.

Algorithm 4 Filter Candidate Goals in Observations.

Input: $\Xi = \langle \Sigma, \mathcal{A} \rangle$ *planning domain*, $\mathcal{I}$ *initial state*, $\mathcal{G}$ *set of candidate goals*, O *observations*, and θ *threshold.*
Output: *A set of filtered candidate goals* $\Lambda_{\mathcal{G}}$ *with the highest percentage of achieved landmarks in observations* O.

```
1: function FilterCandidateGoals($\Xi, \mathcal{I}, \mathcal{G}, O, \theta$)
2:   $\Lambda_{\mathcal{G}} := \langle\rangle$                                  ▷ Map goals to % of achieved landmarks.
3:   for each goal $G$ in $\mathcal{G}$ do
4:     $\mathcal{L}_G :=$ ExtractLandmarks($\Xi, \mathcal{I}, \{G\}$)
5:     $\langle F_{sa}, F_{ua}, F_{st} \rangle :=$ PartitionFacts($\mathcal{L}_G, \mathcal{A}$)   ▷ $F_{sa}$: set of Strictly Activating facts, $F_{ua}$: set of Unstable Activating facts, and $F_{st}$: set of Strictly Terminal facts.
6:     if $F_{sa} \cap \mathcal{I} = \emptyset$ then
7:       continue                                          ▷ Goal G is no longer possible.
8:     $\mathcal{AL}_G := \emptyset$                                    ▷ Achieved fact landmarks for G.
9:     $\mathcal{L}_{\mathcal{I}} :=$ all landmarks $L \in \mathcal{I}$
10:    for each observed action $o$ in $O$ do
11:      if $(F_{ua} \cup F_{st}) \subseteq (pre(o) \cup eff(o)^+ \cup eff(o)^-)$ then
12:        $discardG =$ true
13:        break
14:      else
15:        $\mathcal{L} :=$ remove all landmarks $L$ in $\mathcal{L}_G$ s.t $L \in eff(o)^-$ and $L \in \mathcal{L}$
16:        $\mathcal{L} :=$ all landmarks $L$ in $\mathcal{L}_G$ s.t $L \in pre(o) \cup eff(o)^+$ and $L \notin \mathcal{L}$
17:        $\mathcal{L}_{\prec} :=$ predecessors $L_{\prec}$ of all $L$ in $\mathcal{L}$, s.t $L_{\prec} \notin \mathcal{L}$
18:        $\mathcal{AL}_G := \mathcal{AL}_G \cup \langle \mathcal{L}_{\mathcal{I}} \cup \mathcal{L} \cup \mathcal{L}_{\prec} \rangle$
19:    if $discardG$ then
20:      break                                  ▷ Avoid computing achieved landmarks for G.
21:    $\Lambda_{\mathcal{G}} := \Lambda_{\mathcal{G}} \cup \langle G, \left(\frac{|\mathcal{AL}_G|}{|\mathcal{L}_G|}\right) \rangle$                 ▷ % of achieved landmarks for G.
22:  return all $G$ s.t $\langle G, v \rangle \in \Lambda_{\mathcal{G}}$ and $v \geq (\max_{v_i} \langle G', v_i \rangle \in \Lambda_{\mathcal{G}}) - \theta$
```

Example 6. Consider the Blocks-World example shown in Fig. 2 and that the following actions have been observed in the plan execution: `(unstack E A)` and `(stack E D)`. Using $\theta = 0$, Algorithm 4 returns just the goal `RED` because this goal has achieved 6 out of 10 fact landmarks, so it is the goal in the set of candidate goals with the highest percentage of achieved landmarks in observations. From the first observed action `(unstack E A)`, the algorithm computes in its preconditions the following fact landmark:

- *fact landmarks in preconditions*: `[(on E A) (clear E) (handempty)]`;

Subsequently, the second observed action `(stack E D)` has in its preconditions and effects the following fact landmarks:

- *fact landmarks in preconditions*: `[(clear D) (holding E)]`; and
- *fact landmarks in effects*: `[(on E D)]` (*which is also a sub-goal*);

From the initial state, it is also possible to compute the following set of achieved fact landmarks:

- `[(clear R) (ontable R) (handempty)]`;
- `[(clear D) (on D B) (handempty)]`;
- `[(clear R)]` (*which is also a sub-goal*);

Thus, the estimated percentage of achieved fact landmarks for the goal `RED` is 60%, because it has achieved 6 out of 10 fact landmarks. Note that we consider sub-goals like `(clear R)` and `(on E D)` as independent fact landmarks. Although all sub-goals of a goal must be true together for achieving the goal, in our filtering method we use them separately to estimate the percentage of achieved landmarks.

By contrast, for goals `BED` and `SAD`, the observed actions allow the filtering method to conclude that, respectively, 5 out of 10 and 5 out of 11 fact landmarks have been achieved for these goals. Thus, the estimated percentage of achieved fact landmarks for the `BED` is 50%, and for `SAD` is 45%. From the evidence of fact landmarks in observations `(unstack E A)` and `(stack E D)`, Figs. 5, 6, and 7 show the achieved fact landmarks for the candidate goals `RED`, `BED`, and `SAD`. Boxes in dark gray denote fact landmarks that have been achieved in observations.

Example 6 does not show the real impact of using the set of partition facts (Section 3.2) in our filtering method. However, we argue that the evidence of such partitions in observations can immediately prune impossible candidate goals,

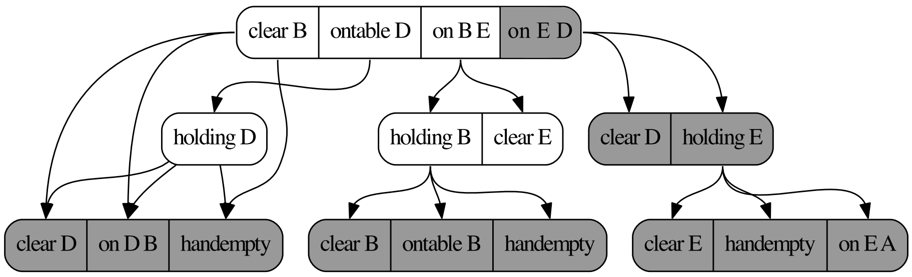


Fig. 6. Fact landmarks for the word BED. Boxes in dark gray show achieved fact landmarks from the observed actions `(unstack E A)` and `(stack E D)`.

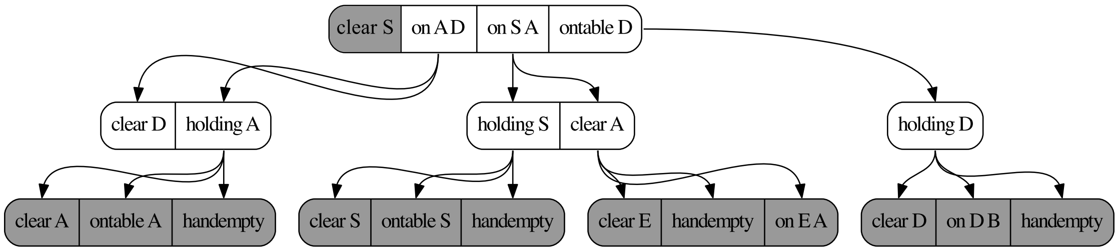


Fig. 7. Fact landmarks for the word SAD. Boxes in dark gray show achieved fact landmarks from the observed actions `(unstack E A)` and `(stack E D)`.

avoiding the computation of achieved landmarks for such goal, improving the recognition time. We also show in Section 5 that our filtering method can be used with other planning-based goal and plan recognition approaches [14,15,19], improving significantly the recognition time for all domains and problems, by reducing the number of calls to a planner (or heuristic).

The complexity of filtering goals (Algorithm 4) in the worst case, including the process of extracting landmarks (EL) and fact partitions (FP) is: $O(|\mathcal{G}| \cdot EL \cdot FP \cdot |O| \cdot |\mathcal{L}_{\mathcal{G}}|)$, where $\mathcal{G}$ is the set of candidate goals, O is the observation sequence, and $\mathcal{L}_{\mathcal{G}}$ is the extracted landmarks for $\mathcal{G}$. Classifying facts into partitions is a simple iteration over the set of actions $\mathcal{A}$.

Proposition 2 *(Soundness of the goal filtering algorithm). Let $T_{GR} = \langle \Xi, \mathcal{I}, \mathcal{G}, O \rangle$ be a goal recognition problem with candidate goals $\mathcal{G}$, a complete and noiseless observation sequence $O = \langle o_1, o_2, ..., o_n \rangle$. If $G^* \in \mathcal{G}$ is the correct hidden goal, then, for any landmark extraction algorithm that generates fact landmarks $\mathcal{L}_{\mathcal{G}}$ and computed landmarks $\Lambda_{\mathcal{G}}$, function* FilterCandidateGoals*($\Xi, \mathcal{I}, \mathcal{G}, O, \theta$) never filters out G^* for any threshold value θ.*

Proof. The proof of this proposition depends on two conditions: first that the reasoning performed over fact partitions never discards G^*; and second, that the ranking using the percentage of achieved landmarks always ranks G^* highest (with possible ties).

The first property then relies on three conditions, namely that we never discard the true goal reasoning about: *Strictly Activating* facts F_{sa} (from Definition 13), *Unstable Activating* facts F_{ua} (from Definition 14), and *Strictly Terminal* facts F_{st} (from Definition 15). Since we only eliminate goals whose landmarks are *Strictly Activating* (F_{sa}) that **are not** in the initial state $\mathcal{I}$, this condition only eliminates goals for which there is no possible plan from the initial state. By Definition 8, O must correspond to a plan from $\mathcal{I}$ that achieves G^*, so, if any landmark of G^* is *Strictly Activating*, it must be in $\mathcal{I}$. Similarly, we only eliminate goals whose landmarks $\mathcal{L}$ are *Unstable Activating* (F_{ua}) and *Strictly Terminal* (F_{st}) if they are part of the preconditions or effects of the observations that occur before l is needed (i.e., that they have become false throughout the plan before they were needed). Since, landmarks $\mathcal{L}$ are necessary conditions, then any valid plan from $\mathcal{I}$ to G^* must only delete $\mathcal{L}$ after they are needed, and thus only goals $G \in \mathcal{G}$ for which observation O is not a valid plan can be discarded. The second condition follows from Theorem 1. □

As a consequence of Proposition 2, any goal recognition algorithm using the filtering mechanism does no worse than the same goal recognition algorithm which does not use the filtering mechanism under full observability. Indeed the empirical results of Section 5.4 corroborate this theoretical result, as the performance for the filtered version of the Ramírez and Geffner [14] algorithm is strictly superior to the algorithm alone for full observability.

5. Experiments and evaluation

In this section, we describe the experiments and evaluation we carried out on our goal recognition approaches. We start with a description of the planning domains and the datasets in Section 5.1, as well as the metrics we use to evaluate our approaches in Section 5.2. Section 5.3 describes the different plan recognition techniques used in the evaluation, and Section 5.4 then details the experiments and evaluation results of our goal recognition approaches.

5.1. Domains and datasets

We empirically evaluated our goal recognition approaches using 15 domains from the planning literature, such as BLOCKS-WORLD, DEPOTS, LOGISTICS, ZENO-TRAVEL, etc. Six of these domains are originally also used in the evaluation of other goal and plan recognition approaches in the literature [14,15].[7]

We formalize planning domains and problems using the STRIPS [28] fragment of PDDL [29]. Based on the datasets provided by Ramírez and Geffner [14,15], which contain hundreds of goal recognition problems for 6 domains, we added non-trivial[8] and larger planning problems in their datasets and generated new datasets[9] from the remaining 9 planning domains using open-source planners, such as FAST-DOWNWARD [41], FAST-FORWARD [39], and LAMA [35], 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 approaches [42]. We also generated datasets for 4 domains (CAMPUS, INTRUSION, IPC-GRID, and KITCHEN) with missing, full, and noisy observations, which are the same domains that Sohrabi et al. use in [19]. The dataset for CAMPUS domain with missing and noisy observations comes from Sohrabi et al. [19].[10] Thus, we evaluate our goal recognition approaches against the state-of-the-art not only using datasets with missing and full observations, but also using datasets with noisy observations in the same way as Sohrabi et al. [19] use.

5.2. Evaluation metrics

For evaluation of our goal recognition approaches against the state-of-the-art, we use the *Accuracy* metric (Equation (4)), which represents the fraction of problems in which the correct goal was among the goals found to be most likely, i.e., how well the correct hidden goal is recognized from a set of possible goals for a given goal recognition problem. Most goal recognition approaches [15,18,19] refer to this metric as quality, also denoted as Q.

$$\textsc{Accuracy} = \frac{\sum True\ \ positive + \sum True\ \ negative}{\sum Total\ \ population} \tag{4}$$

Like most goal recognition approaches in the literature, we also evaluate the average number of returned goals (i.e., the most likely goals), which is called the *Spread* in $\mathcal{G}$, as well as the recognition time (speed) in seconds, representing the time that a goal recognition approach takes to recognize the most likely goal from a set of possible goals.

Generally, we compute the *Accuracy* and *Spread* in $\mathcal{G}$ metrics very similarly for both probabilistic and score-based approaches. For score-based approaches, we consider the correct goal to have been selected if it is one of the goals tied with the highest score, whereas for probabilistic approaches, we consider the correct goal to have been selected if it is either the single goal with the highest probability, or if the correct goal is tied with the highest probability with any other goal.

5.3. Evaluated approaches

Our experiments compare our recognition heuristics (h_{gc} and h_{uniq}) and the use of our filtering method to the five state-of-the-art approaches in goal and plan recognition as planning, as follows.

R&G 2009: We use the recognition approach of Ramírez and Geffner [14] based on a heuristic estimator, denoted as R&G 2009; as well as a combination of their approach and our filtering method with threshold $\theta = 10\%$, denoted as $\textsc{Filter}_{10\%}+$ R&G 2009. This recognition approach uses a heuristic method to approximate the planning solution by computing a relaxed plan [43]. The authors show that this heuristic-based approach is their faster and more accurate goal and plan recognition approach. This is their fastest and most accurate approach.

R&G 2010: We use the probabilistic framework proposed by Ramírez and Geffner [15] that allows the use of any off-the-shelf automated planner, denoted as R&G 2010. The automated planner we used alongside R&G 2010's framework is FAST-DOWNWARD [41] with the LM-Cut heuristic [44], a planning heuristic that relies on landmarks to estimate the distance

[7] https://sites.google.com/site/prasplanning/.

[8] A non-trivial planning problem contains a large search space (in terms of search branching factor and depth), and therefore, modern planners such as FAST-DOWNWARD take up to 5 minutes to solve it. In our datasets, the number of instantiated (grounded) actions is between 158 and 3258, and plan length is between 5 and 83.

[9] https://github.com/pucrs-automated-planning/goal_plan-recognition-dataset.

[10] https://github.com/shirin888/planrecogasplanning-ijcai16-benchmarks.

from a particular state to a goal state. We also use this approach with our filtering method (threshold $\theta = 10\%$), denoted as FILTER$_{10\%}$+ R&G 2010.
IBM 2016: We use the approach of Sohrabi et al. [19],[11] which uses a top-K planner to extract multiple optimal and nearly optimal plans for a particular goal, denoted as IBM 2016[12] The automated planner we used alongside this approach is the most recent top-K planner TK* [45] with the LM-Cut heuristic, and the number of sampled plans parameter K=1000. These are exactly the same parameters that [19] used in their experiments and evaluation. Note that we use the LM-Cut heuristic [44] with the approaches from [15] and [19] because our own goal recognition approaches rely on landmarks. We use this landmark-based planning heuristic with the FAST-DOWNWARD [41] planner and top-K planner TK* [45] aims to provide a fairer comparison against our landmark-based approaches.
FGR 2015: We compare our approaches against the approach of E-Martín et al. [18], denoted as FGR 2015. Their goal recognition approach also obviates multiple calls to a planner for the recognition process, and instead uses a planning graph resulting in fast goal recognition in planning settings. As advised by the authors, we use the FGR 2015 recognizer with interaction information equal to 0, which is their technique that yields the best results in terms of recognition time and accuracy.[13] In both non-noisy (missing) and noisy domains, the IPC-GRID domain timed out for more than 60% of the problems in the FGR 2015 approach, so we do not report results for this specific domain as they would not be representative.
M+L 2018: We also compare our approaches against the offline mode of a combination of landmarks and Goal Mirroring [24,25], denoted as M+L 2018. In [24,25], Vered et al. develop an online goal recognition approach that combines the use of landmarks and Goal Mirroring for both continuous and discrete domain models. The automated planner we used alongside M+L 2018 is a Java version of FAST-FORWARD [39] (JavaFF), the same planner used by [24,25] in their experiments. For extracting landmarks, they use the same landmark extraction algorithm we used for our recognition heuristics, i.e., the algorithm of Hoffmann et al. [23].

5.4. Goal recognition experimental results

All evaluated approaches take as input a goal recognition problem T_{GR} from Definition 8, i.e., a domain description as well as an initial state, a set of candidate goals $\mathcal{G}$ (a correct hidden goal $G^* \in \mathcal{G}$), and an observation sequence O. An observation sequence contains actions that represent an optimal or sub-optimal plan that achieves a correct hidden goal G^*, and this observation sequence can be full or partial (i.e., with missing observations). Full observation sequences contain the entire plan for a correct hidden goal G^*, i.e., 100% of the actions having been observed. Partial observation sequences with missing actions represent plans for a correct hidden goal G^* with 10%, 30%, 50%, or 70% of their actions having been observed. However, for experiments with noisy observations, the observability of partial observations is quite different because every observation sequence always includes at least two noisy observations. Thus, we start with a higher percentage of observations for noisy domains, with sequences representing a plan with 25%, 50%, or 75% of its actions having been observed.

Our evaluation uses three metrics: accuracy of goal recognition (Equation (4)); the average number of goals in $\mathcal{G}$ that have been found to be the most likely; and recognition time (speed). Note that in many domains, all algorithms return more than one candidate goal. In the case of Ramírez and Geffner [14], i.e., R&G 2009, this may occur when goals have the same distance from their estimated state. Alternatively, for our goal recognition heuristics, this may occur when there are ties between the heuristic value of candidate goals within the threshold margin. Thus, like most goal recognition approaches, we also evaluate the average number of returned goals for a given goal recognition problem, i.e., the spread in $\mathcal{G}$, denoted as S in $\mathcal{G}$. The ideal value for spread in $\mathcal{G}$ (S in $\mathcal{G}$) is 1, i.e., when the recognizer returns only one goal as the correct and intended goal that the observed agent aims to achieve from the observations. We ran all experiments using a single core of a 12 core Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40 GHz with 16 GB of RAM, set a maximum memory usage limit of 4 GB, and set a 20-minute timeout for each recognition process.

5.4.1. Experimental results with missing and full observations

Our first set of experiments consists of running the various goal recognition approaches in datasets containing thousands of problems for 15 planning domains with missing and full observations. For our heuristics h_{gc} and h_{uniq}, we show their results under different thresholds: 0%, 10%, and 20%. If the threshold value is $\theta = 0$, our heuristics do not give any flexibility for estimating candidate goals, returning only the goals with the highest estimated value.

Table 1 summarizes our experimental results presenting a comparison of all evaluated goal recognition approaches, showing averages for all used metrics over all goal recognition problems for 15 planning domains. In what follows, each row in Table 1 expresses, respectively: the average time (in seconds) to recognize the goal given the observations (Time);

[11] Since the exact code from Sohrabi et al. [19] is unavailable, we developed our own version of this approach with some advice from the main author and the top-k planner she shared.

[12] We ran experiments using a top-k planner rather than a diverse planner under advice from the main author of [19].

[13] In an attempt to make a fair comparison, we obtained the code for the algorithms of [18] directly from the main author. Running on our datasets, these algorithms performed worse than the results of [18]. We believe that this could be due to different problem set sizes of our datasets. In addition, the code behaved unexpectedly on some domains of our datasets (denoted by a † symbol in the tables in Appendix A), returning the same recognition score for all candidate goals. At the time of submission, we are working with the authors to clarify these discrepancies.

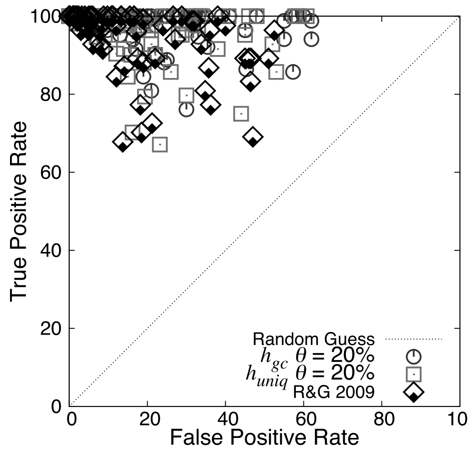
Table 1
A summarization of our experimental results with missing and full observations. The average number of candidate goals ($|\mathcal{G}|$) in the datasets we used is 8.43. The symbols $^\diamond$ and $^\bullet$ indicate that the average number of goals for FGR 2015 and M+L 2018 is different because these approaches timed-out and/or had out of memory for some domains, and thus, the average number of goals for each them is, respectively, 8.81 and 7.36. The symbol * indicates that the IBM 2016's approach timed-out for 14 domains in our datasets (out of 15 domains) so their overall average would be meaningless here. We refer the reader to Table A.4 in Appendix A for that result.

Approach	10%			30%			50%			70%			100%		
	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$
R&G 2009	1.412	92.90%	3.49	1.508	93.52%	2.19	2.034	94.73%	1.81	3.187	94.87%	1.59	5.099	88.16%	1.39
R&G 2009 + Filter$_{10\%}$	0.781	83.48%	2.52	0.944	88.92%	1.54	1.007	93.74%	1.31	1.460	96.96%	1.16	1.603	100.0%	1.09
R&G 2010	547.908	70.24%	3.08	636.196	71.61%	2.04	685.231	70.21%	1.98	713.919	70.39%	2.11	687.844	75.84%	2.38
R&G 2010 + Filter$_{10\%}$	378.675	70.92%	1.52	370.444	74.55%	1.46	330.127	78.46%	1.28	275.836	84.46%	1.17	224.897	88.50%	1.04
FGR 2015$^\diamond$	57.336	82.85%	5.07	50.021	90.37%	5.23	54.677	91.62%	5.21	62.924	94.94%	4.91	79.464	98.25%	3.16
IBM 2016*	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
M+L 2018$^\bullet$	5.548	62.82%	1.63	5.594	82.98%	1.38	5.095	90.47%	1.27	4.940	94.05%	1.19	4.786	94.36%	1.12
h_{gc} ($\theta = 0$)	0.290	50.21%	1.32	0.297	69.99%	1.11	0.312	78.72%	1.07	0.323	90.58%	1.04	0.331	100.0%	1.03
h_{gc} ($\theta = 10$)	0.298	74.23%	2.75	0.303	83.94%	2.09	0.319	89.52%	1.62	0.327	93.43%	1.34	0.338	100.0%	1.15
h_{gc} ($\theta = 20$)	0.304	86.37%	4.01	0.312	90.82%	3.23	0.325	93.04%	2.33	0.333	95.03%	1.77	0.341	100.0%	1.47
h_{uniq} ($\theta = 0$)	0.287	53.07%	1.41	0.291	70.60%	1.24	0.308	81.50%	1.17	0.316	92.27%	1.13	0.320	100.0%	1.03
h_{uniq} ($\theta = 10$)	0.293	76.63%	2.68	0.297	85.92%	2.12	0.311	91.75%	1.79	0.317	97.68%	1.46	0.322	100.0%	1.17
h_{uniq} ($\theta = 20$)	0.299	89.72%	3.69	0.322	94.03%	3.02	0.328	97.01%	2.30	0.336	99.10%	1.89	0.338	100.0%	1.39

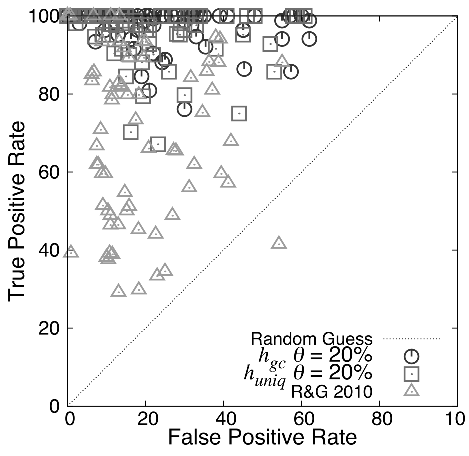
the accuracy (Acc %) with which the approaches correctly infer the goal; and the spread in $\mathcal{G}$, which represents the average number of returned goals (S in $\mathcal{G}$). Apart from our recognition heuristics and the approaches of Ramírez and Geffner (2009 and 2010), all other evaluated approaches have faced some issues with some domains and recognition problems, more specifically, parsing problems, out of memory errors, timeouts, and other issues. FGR faced some issues with 5 domains in the datasets we used, timing out for all problems for IPC-Grid, and returning the same probability value for all goals for all problems for Depots, Logistics, Miconic, and Rovers. As mentioned before, IBM 2016 timed out for most problems in our datasets (we set a 20-minute timeout for every recognition problem). As for M+L 2018, this approach ran out of memory (we set a maximum memory usage limit of 4GB for every recognition problem) for 4 domains in our datasets, namely Depots, Driver-Log, DWR, and Sokoban.

The average recognition time for our approaches are significantly faster than all previous evaluated approaches for all domains and problems. For instance, for all levels of observability (from 10% to 100%), our approaches are $\approx$ 8.6 times faster than R&G 2009, $\approx$ 2063.8 times faster than R&G 2010, $\approx$ 197.7 times faster than FGR 2015, and $\approx$ 16.5 times faster than M+L 2018. Our filtering method also significantly improves the recognition time for R&G 2009 and 2010 by more than 50%. Note that the recognition time reported in Table 1 for our approaches (heuristics and filter) includes the landmark extraction process.

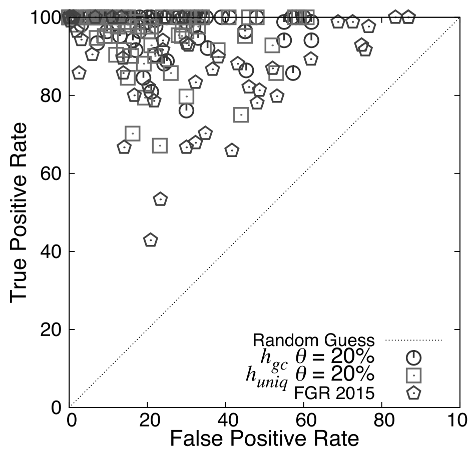
The averages for accuracy (Acc %) and spread (S in $\mathcal{G}$) for our recognition heuristics in Table 1 are, in general, very competitive compared to all evaluated approaches, and for some levels of observability significantly better than the other approaches. In comparison to the approaches that did not time out or had issues with some recognition problems — specifically R&G 2009 and 2010 — our heuristics show competitive results against these approaches regarding accuracy and spread in $\mathcal{G}$, especially when we increase the threshold value θ to 10% and 20%. For low levels of observability (from 10% to 50%) our heuristics are slightly worse than R&G 2009 and 2010, while being slightly better for high levels of observability (70% and 100%). Note that the use of our filtering method reduces the spread in $\mathcal{G}$ for both R&G 2009 and 2010, however, it slightly reduces accuracy for most levels of observability (from 10% to 50%), improving accuracy and reducing spread when we have more observations (i.e., for 70% and 100% of observability). The average results show that our filtering method provides not only substantial gains in recognition time for R&G 2009 and 2010 (50% gain in recognition time as shown in Table 1), but also some gain in accuracy and spread for high levels of observability (70% and 100%). Note that, while the accuracy of the FGR 2015 approach is high ($\geq$ 82.5%) for all levels of observability, the spread in $\mathcal{G}$ is also quite high, returning as the correct intended goal on average more than $\approx$ 4.5 goals (out of 8.81 goals) for all levels of observability. This means that while the correct goal is included in the recognition, the number of spurious goals included together with the correct goal is also very high, resulting in low precision. Table 1 shows that the results for our heuristics are better than FGR 2015 for all evaluated metrics when the threshold θ is higher than 0%, i.e., $\theta = 10\%$ and $\theta = 20\%$, showing that our heuristics are more precise when we increase the threshold value θ. While the results for M+L 2018 approach surpass the results of our heuristics for both accuracy and spread in $\mathcal{G}$, M+L 2018 ran out of memory for 4 domains in our datasets, whereas our heuristics did not. When using the threshold value $\theta = 10\%$ for both heuristics, our results for accuracy and spread in $\mathcal{G}$ are very competitive compared to M+L 2018, and always better for 100% of observability. Similar to our filtering method, M+L 2018 uses landmarks to rule out candidate goals before using the Goal Mirroring approach, however, this approach is not as fast as our filtering method alongside R&G 2009 and 2010. We show graphically in Fig. 8 the trade-off between true positive (i.e., accuracy) and false positive results of our heuristics compared to all evaluated approaches.



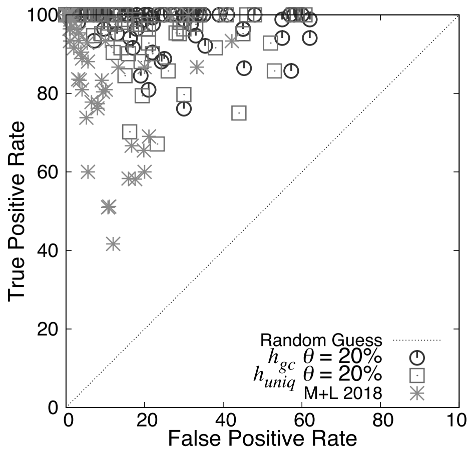
(a) h_{gc}, h_{uniq}, and R&G 2009.



(b) h_{gc}, h_{uniq}, and R&G 2010.



(c) h_{gc}, h_{uniq}, and FGR 2015.



(d) h_{gc}, h_{uniq}, and M+L 2018.

Fig. 8. ROC space aggregating the results of all domains with missing and full observations comparing our landmark-based heuristics (h_{gc} and h_{uniq}) against R&G 2009 [14], R&G 2010 [15], FGR 2015 [18], and M+L 2018 [24]. The results obtained for CAMPUS domain from IBM 2016 [19] are not included in the ROC space.

In Appendix A, we detail our experimental results for missing and full observations, showing all results separately for each of the 15 planning domains and all evaluated approaches. Tables A.3 and A.4 show comparative results of our heuristics and previous approaches for the first set of domains (from BLOCKS-WORLD to INTRUSION). Table A.3 shows the results of our goal recognition heuristics h_{gc} and h_{uniq}, against R&G 2009 [14] as well as this approach enhanced with our filtering method with threshold $\theta = 10\%$ (Filter$_{10\%}$). Similarly, Table A.4 shows the results of R&G 2010 [15], FGR 2015 [18], IBM 2016 [19], and M+L 2018 [24,25]. Tables A.5 and A.6 show comparative results of our heuristics and previous approaches for the second set of domains (from KITCHEN to ZENO-TRAVEL). From these tables, it is possible to see that our heuristics are both faster and more accurate than R&G 2009, R&G 2010, FGR 2015, IBM 2016, and M+L 2018. Note that the resulting combination of our filtering method and the approaches of R&G (2009 and 2010) get a substantial speedup and often accuracy improvements. As we increase the threshold, our heuristic approaches quickly surpass all approaches in almost all domains we tested. For more details, we refer the reader to Tables A.3, A.4, A.5, and A.6.

Experimental results: ROC space

We adapt the *Receiver Operating Characteristic (ROC)* curve metric to show the trade-off between true positive and false positive results. The ROC curve allows us to provide a summary of the discriminatory performance of inferences such as goal recognition over diverse datasets. The ROC curve graphically shows the performance of classifier systems by evaluating true positive rate against the false positive rate at various threshold settings. We adapt the notion of the ROC curve into points over the ROC space to compare not only true positive predictions (i.e., Accuracy), but also to compare the false positive ratio of the goals recognition approaches we evaluated. Each prediction result of a goal recognition approach represents one point in the ROC space. In this space, the diagonal line represents a random guess to recognize a goal from observations. This line thus divides the ROC space, points above the diagonal represent better than random classification results (i.e., "good"),

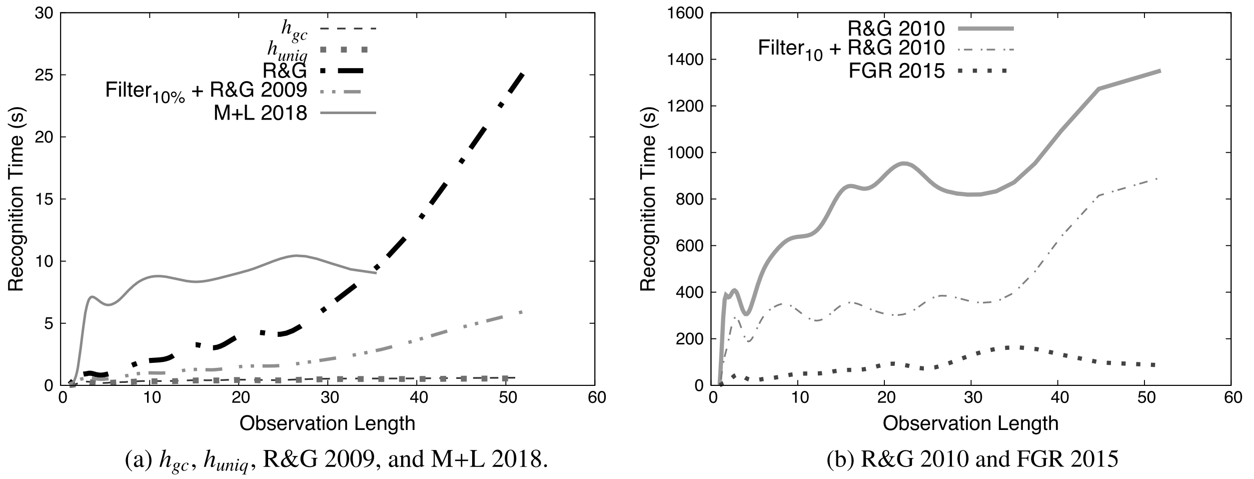


Fig. 9. Recognition time comparison for missing and full observations for our landmark-based heuristics (h_{gc} and h_{uniq}) against R&G 2009 [14], R&G 2009 using our filtering method with 10% of threshold, R&G 2010 using FAST-DOWNWARD with LM-Cut heuristic [15], R&G 2010 using our filtering method with 10% of threshold, FGR 2015 [18], and M+L 2018 [24] using FAST-FORWARD with EHC heuristic. The recognition time for our heuristics and the approaches that use our filtering method include the time to extract landmarks.

whereas points below the line represent worse than random results (i.e., “poor”). The best possible (perfect) prediction for recognizing goals is a point in the upper left corner (i.e., coordinate $x = 0$ and $y = 100$) of the ROC space. Thus, the closer a goal recognition approach (point) gets to the upper left corner, the better it is for recognizing goals. To visualize the comparative performance of the multiple approaches, we adapt the notation of the ROC space, and, rather than plotting a single point per goal recognition problem, we aggregate multiple problems for all domains and plot these results in ROC space.

Fig. 8 shows the trade-off between true positive results and false positive results in ROC space for the evaluated goal and plan recognition approaches. Recall that the closer a goal recognition approach (point) is to the upper left corner, the better it is for recognizing goals and plans. To compare the recognition results of our approaches against the others in the ROC space, we select the results of our heuristics using the threshold $\theta = 20\%$. 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 hidden goal from missing and full observations. The ROC space plots show that our heuristics are not only competitive against the other five approaches (R&G 2009, R&G 2010, FGR 2015, IBM 2016, M+L 2018) for all variations of observability, but also surpasses these approaches in a substantial number of domains and problems. Fig. 8a shows a comparison of our two heuristics against R&G 2009, and it is possible to see that all three compared approaches have most points in the upper left corner, showing that these approaches have similar results (R&G 2009 being slightly worse than our heuristics) regarding true and false positive results. When comparing our heuristics against R&G 2010 (Fig. 8b), note that most points for R&G 2010 are not closer to the upper left corner, showing that R&G 2010 has poor results (low true positive rate or accuracy) for most goal recognition problems. As for FGR 2015, this approach has high true positive results for most points, but also has high false positive results, having several points close to the random guess (Fig. 8c), while our heuristics have most points close to the upper left corner. With respect to M+L 2018, it has competitive results in comparison to our heuristics (Fig. 8d), however, M+L 2018 has some points with low true positive rate, indicating that some problems in some domains prove challenging for it.

Experimental results: recognition time

We compare the time that each approach takes to recognize the correct hidden goal for different sizes of the observation sequence. Figs. 9a and 9b show the runtime as a function of the average length of the observation sequences ($|O|$ column in Tables A.3–A.6) for all of the approaches we evaluated (apart from IBM 2016, which timed out for almost all domains), as reported in the time column of Tables A.3–A.6). We use separate graphs for R&G 2010, Filter$_{10\%}$+ R&G 2010, and FGR 2015 given the widely different magnitude of the time taken to recognize a goal. Fig. 9a shows the runtime for our heuristic approaches in comparison with R&G 2009 and this approach alongside our filtering method, whereas Fig. 9b shows the runtime of R&G 2010 with and without the filtering method, FGR 2015, and M+L 2018. When measuring recognition time for our heuristics and the filtering method, we also include the time to extract the set of landmarks, so that landmark extraction is performed online, i.e., during the goal recognition process. Curves in the graph represent the average runtime when observation sizes were the same smoothed over the resulting points. Our goal recognition heuristics never take more than 1 second (≈ 0.7 seconds) to compute the correct hidden goal in the set of candidate goals, while the other approaches seem to grow super-linearly (for R&G 2009), and exponentially (for R&G 2010). The two lines (- - - and $\cdots\cdots$) for our heuristics overlap because the recognition time for both is very similar over the datasets, but from the recognition time reported in the tables, we can see that the uniqueness heuristic h_{uniq} ($\cdots\cdots$ line) is slightly faster than h_{gc} (- - - line).

Table 2
Experiments and evaluation with missing, noisy, and full observations.

Approach	25%			50%			75%			100%		
	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$
R&G 2009	0.235	60.71%	3.27	0.244	61.93%	3.01	0.267	63.79%	2.87	0.338	64.79%	2.86
R&G 2009 + Filter$_{10\%}$	0.223	53.74%	1.94	0.236	58.04%	1.30	0.262	60.21%	1.15	0.314	65.48%	1.05
R&G 2010	9.872	46.54%	1.10	11.663	55.31%	1.10	12.674	62.31%	1.08	13.405	70.64%	1.13
R&G 2010 + Filter$_{10\%}$	6.916	48.68%	1.0	3.361	55.31%	1.0	2.491	61.21%	1.0	2.635	68.97%	1.0
FG 2015$^{\diamond}$	0.524	58.84%	1.66	0.529	74.31%	1.56	0.578	80.41%	1.37	0.693	90.34%	1.38
IBM 2016*	–	–	–	–	–	–	–	–	–	–	–	–
M+L 2018	0.199	53.81%	1.45	0.155	71.89%	1.30	0.163	79.45%	1.26	0.149	73.24%	1.06
h_{gc} $(\theta = 0)$	0.083	42.89%	1.03	0.089	68.44%	1.01	0.097	71.74%	1.04	0.101	83.02%	1.08
h_{gc} $(\theta = 10)$	0.086	61.37%	2.22	0.090	75.95%	1.41	0.099	78.48%	1.11	0.115	88.84%	1.07
h_{uniq} $(\theta = 0)$	0.080	63.60%	1.63	0.084	72.63%	1.27	0.095	78.41%	1.20	0.096	83.02%	1.03
h_{uniq} $(\theta = 10)$	0.081	78.51%	2.70	0.086	84.01%	1.89	0.095	85.70%	1.52	0.096	89.67%	1.30

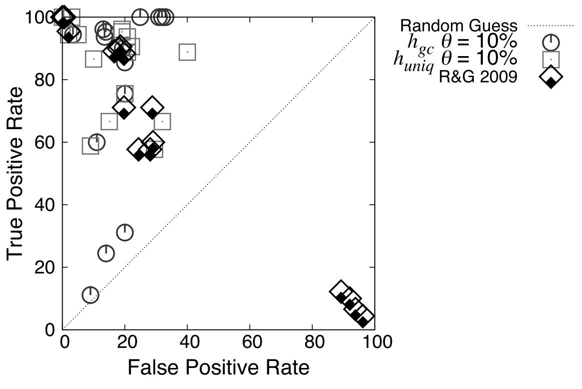
The approaches of Ramírez and Geffner [14,15], R&G 2009 and R&G 2010, took at most $\approx$ 25 seconds and $\approx$ 1200 seconds, respectively. Apart from the Campus domain, the approach of IBM 2016 [19] timed out for all goal recognition problems (all domains) in the datasets we used, probably because the top-K planner [45] (even the latest top-K planning algorithm) does not scale very well when dealing with non-trivial planning problems, especially when the planner has to sample 1000 plans for a (transformed) planning problem. In this case, even the use of our filtering method (which reduces the number of candidate goals) did not improve the recognition time of the approach from IBM 2016 [19]. While our filtering method significantly improves the recognition time of the approaches R&G 2009 and R&G 2010, it sometimes causes a loss of accuracy due to it ruling out the correct hidden goal from the set of candidate goals. FGR 2015 took at most $\approx$ 355 seconds over all evaluated domains (and timed out for most problems of the IPC-Grid domain). Fig. 9a shows that M+L 2018 took at most $\approx$ 10 seconds over all evaluated domains, however, the recognition time curve for this approach goes until observation length equals to 36, and it has happened due to the fact that this approach ran out of memory for 4 domains (Depots, Driver-Log, DWR, and Sokoban). M+L 2018 is much faster than FGR 2015 and R&G 2010, and R&G 2010 with our filtering method, though not as fast as our recognition heuristics and R&G 2009. Finally, the evaluation of the domains DWR and Sokoban shows that larger plan and observation lengths lead R&G 2009 and R&G 2010 to rapidly lose accuracy, and M+L 2018 to run out of memory, whereas our approaches show improved accuracy without affecting the recognition time (of at most $\approx$ 0.7 seconds).

5.4.2. Experimental results with missing, noisy, and full observations

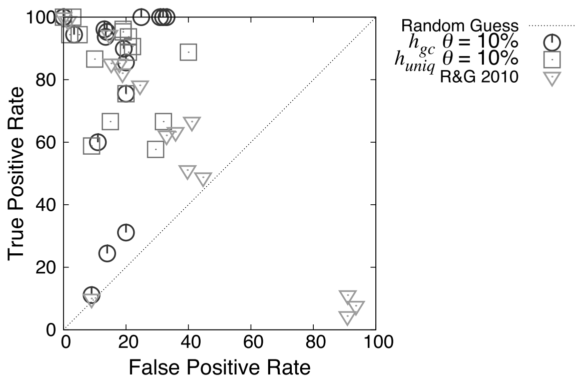
Our second set of experiments uses datasets containing hundreds of problems for four domains with missing, noisy, and full observations. We compare results for the experiments using these datasets for our goal recognition heuristics (using a threshold between 0% and 10%) against R&G 2009 [14], R&G 2010 [15], FGR 2015 [18], and IBM 2016 [19], and M+L 2018 [24]. We use our filtering method with a 10% of threshold alongside two other approaches (R&G 2009 and 2010), denoted as Filter$_{10\%}$. For this set of experiments, we build datasets using the same 4 domains (Campus, Intrusion, IPC-Grid, and Kitchen) used by Sohrabi et al. [19]. In these experiments, every observation sequence O has two actions as noisy observations, i.e., two extra observations that we added randomly to the observation sequence O. These two extra observations represent 12% noise in relation to the total number of observations [19]. Since Sohrabi et al. [19] timed out for all recognition problems, we are unable to provide a direct comparison of accuracy and runtime performance against this approach. However, given our understanding of the underlying technique, we believe that our approaches are almost certainly more computationally efficient, since they use a top-K planner [45] during the recognition process (extracting 1000 sampled plans), much like [15] that uses a planner for recognizing goals.

Table 2 summarizes the experimental results under noisy observations for all evaluated approaches considering four levels of observability, i.e., 25%, 50%, 75%, and 100%. The R&G 2009 and R&G 2010 approaches are not only much slower but substantially less accurate (with a $\theta = 10\%$ threshold value) than our heuristics for virtually all 4 domains. Our recognition heuristics are $\approx$ 2.7 times (on average) faster than R&G 2009 and R&G 2010, even when using our filtering method. However, when we use the threshold value $\theta = 0\%$, the R&G 2009 and R&G 2010 approaches are more accurate than our heuristics for all levels of observability. While the results for FGR 2015 are similar to ours for accuracy and spread in $\mathcal{G}$ using $\theta = 0\%$, our approaches are still $\approx$ 6.1 times faster on average than FGR 2015. Note that when increasing the threshold value θ for our heuristics, the results for spread in $\mathcal{G}$ also increase, obtaining closer values to FGR 2015, reaching higher values for accuracy for all levels of observability. Comparing the accuracy and spread in $\mathcal{G}$ of our approach (as well as R&G 2010, and FGR 2015) to M+L 2018 under noisy observations, M+L 2018 shows similar and competitive results, and performs slightly better than R&G 2009 when considering the spread, which is on average 50% lower than the values of R&G. M+L 2018 also shows competitive results regarding recognition time against our heuristics, being slightly slower than our heuristics for larger observation lengths (see Fig. 11a, — line). Specifically, our heuristics are $\approx$ 1.8 times faster than M+L 2018.

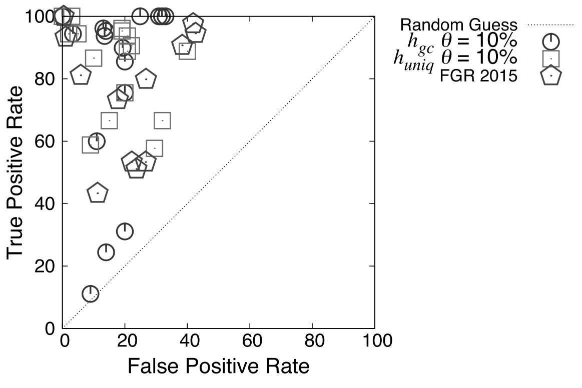
Fig. 10 shows the trade-off between true positive results and false positive results in a ROC space for all 4 domains with missing, noisy, and full observations. Figs. 11a and 11b show a comparison of recognition time for our heuristics against all evaluated approaches. We also use separate graphs for R&G 2010, Filter$_{10\%}$+ R&G 2010, and FGR 2015 given the widely different magnitude of the time taken to recognize a goal. In Appendix B, we detail our experimental results for noisy, missing, and full observations, showing all results separately for each of the 4 planning domains and all evaluated approaches.



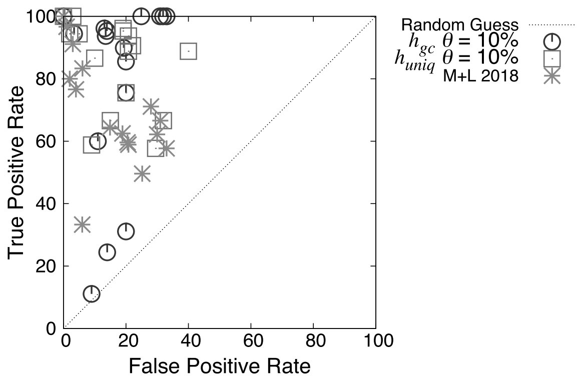
(a) h_{gc}, h_{uniq}, and R&G 2009.



(b) h_{gc}, h_{uniq}, and R&G 2010.

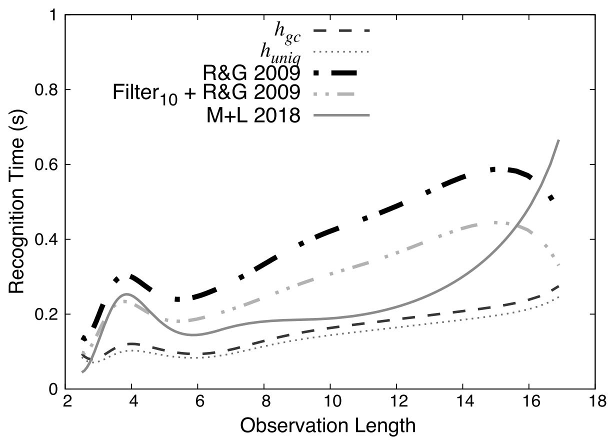


(c) h_{gc}, h_{uniq}, and FGR 2015.

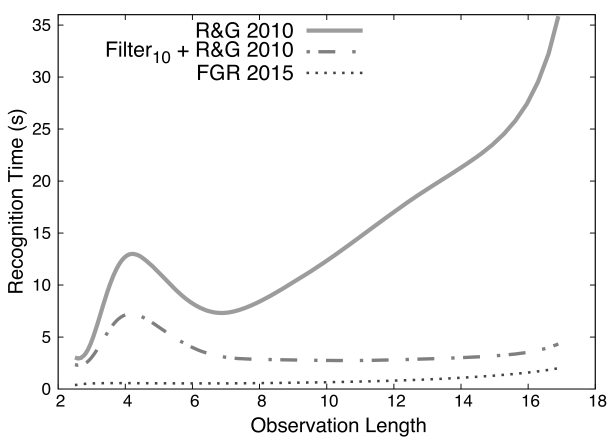


(d) h_{gc}, h_{uniq}, and M+L 2018.

Fig. 10. ROC space for all domains with missing, noisy, and full observations for our landmark-based heuristics (h_{gc} and h_{uniq}) against R&G 2009 [14], R&G 2010 [15], FGR 2015 [18], and M+L 2018 [24].



(a) h_{gc}, h_{uniq}, R&G 2009, and M+L 2018.



(b) R&G 2010 and FGR 2015

Fig. 11. Recognition time comparison for missing, noisy, and full observations for our landmark-based heuristics (h_{gc} and h_{uniq}) against R&G 2009 [14], R&G 2009 using our filtering method with 10% of threshold, R&G 2010 using FAST-DOWNWARD with LM-Cut heuristic [15], R&G 2010 using our filtering method with 10% of threshold, FGR 2015 [18], and M+L 2018 [24] using FAST-FORWARD with EHC heuristic. The recognition time for our heuristics and the approaches that use our filtering method include the time to extract landmarks.

6. Related work

We now compare our work to some of the most relevant recent work on goal and plan recognition in recent years, highlighting differences and similarities between our approaches and the surveyed related work.

Hong [46] developed one of the first goal recognition approaches that extends the concept of a planning graph (which they call a goal graph), developing a similar structure that represents every possible path (i.e., state transitions that connect facts and actions) from an initial state to a goal state. Pattison and Long [16] 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 [14] developed 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. We compare their most accurate approach directly with ours. Follow-up work [15] extended the idea of plan recognition as planning into a probabilistic approach using off-the-shelf planners that provide a posterior probability distribution over goals, given an observation sequence as evidence. E-Martín et al. [18] developed a planning-based goal recognition approach that propagates cost and interaction information in a planning graph, and uses this information to estimate goal probabilities over the set of candidate goals. Sohrabi et al. [19] developed 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. Nevertheless, our heuristic computation is a good proxy for the posterior probability distribution of the goals, given the observations, and thus could be extended to provide a probabilistic interpretation as we intend to do in future work. In [47], Vered et al. introduce the concept of mirroring to develop an online goal recognition approach for continuous domains. Masters and Sardina [22] propose a fast and accurate goal recognition approach that works strictly in the context of path-planning, providing a new probabilistic framework for goal recognition in path planning. In [48], Vered and Kaminka develop a heuristic approach for online goal recognition that deals with continuous domains. Vered et al. [24] propose an online goal recognition approach that combines the use of landmarks and goal mirroring, showing this combination can improve not only the recognition time, but also the accuracy for recognizing goals in the online fashion. In this work, we compare our approaches against this approach [24] in offline mode. Most recently, Kaminka et al. [49] develop a novel plan recognition approach that deals with both continuous and discrete domains.

Secondly, there has been substantial recent work on goal and plan recognition *design*, that is optimized the domain design so that goal and plan recognition algorithms can provide inferences with as few observations as possible. Keren et al. [17,50,51] develop an alternate view of the goal recognition problem, and rather than developing new goal recognition algorithms, they develop a novel approach that modifies the domain description in order to facilitate the goal recognition process. Their work could potentially be used alongside our techniques, and the relation between worst case distinctiveness (their measure of how difficult can it be to disambiguate goals) and the information gain from unique landmarks would provide an interesting avenue for further investigation.

Finally, Freedman et al. [52] recently proposed an approach to perform probabilistic plan recognition along the lines of [15], which, instead of running a full-fledged planner for each goal, takes advantage of multiple-goal heuristic search [53] to search for all goals simultaneously and avoid repeatedly expanding the same nodes. Their approach has not been implemented and evaluated yet and it aims to overcome the limitation of our technique to only be able to account for progress towards goals when we have evidence of landmarks being achieved, while retaining the speed gains we achieve. While we do not have empirical evidence about its accuracy and efficiency, we believe this is an exciting direction for goal recognition, and we expect it to perform similarly to, and perhaps exceed the accuracy of [15].

7. Conclusions

In this work we introduced novel goal recognition approaches based on planning techniques that rely on landmarks. Landmarks provide key information about what cannot be avoided to achieve a goal, and we have shown that they can be used to efficiently build simple heuristics to recognize goals from missing and noisy observations. Our goal completion heuristic h_{gc} computes the ratio between achieved landmarks and the total number of landmarks for a particular goal, whereas our uniqueness heuristic h_{uniq}, uses a *landmark uniqueness value* to represent how informative a landmark is among the known landmarks for all candidate goals. These landmark-based recognition heuristics show that it is possible to recognize goals quickly with high accuracy, and to use them as a filtering mechanism to refine existing planning-based goal and plan recognition approaches [14,15,18,19,24], so that they can be made substantially more efficient.

We have proved that our heuristic approaches are sound both as a filtering mechanism and as a goal recognition algorithm on its own, showing that, under certain conditions, we are guaranteed to find the correct hidden goal. Our experiments show that our goal recognition approaches yield not only superior accuracy results but also substantially faster recognition time for all fifteen planning domains used in evaluating against the state-of-the-art [14,15,18,19,24] at varying observation completeness levels, for both missing and noisy observations. The main limitation of our approaches lie in conditions of very low observability ($< 30\%$). Specifically, for problems with very short plans, and thus, where the number of actually observed action consists of one or two actions, the odds of observing one of the problem's landmarks are very low, jeopardizing recognition accuracy. Under these conditions, our filtering mechanism still provides a major improvement on the runtime (and often accuracy) of existing goal recognition approaches.

As future work, we intend to explore multiple avenues to improve our goal recognition heuristic approaches. First, we aim to use other planning techniques, such as traps, invariants, and dead-ends [54]. Second, we intend to explore other landmark extraction algorithms to obtain additional information from planning domains [37,38]. Third, we aim to evaluate our landmark-based heuristics for online goal and plan recognition, and we have started work in that direction in [24].

Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgements

This article is a revised and extended version of two papers published at AAAI 2017 [21] and ECAI 2016 [20]. We thank the anonymous reviewers that helped improve the research in this article. The authors thank Shirin Sohrabi for discussing the way in which the algorithms of [19] should be configured, and Yolanda Escudero-Martín for providing code for the approach of [18] and engaging with us. We also thank Miquel Ramírez and Mor Vered for various discussions, and André Grahl Pereira for a discussion of properties of our algorithm. Felipe thanks CNPq for partial financial support under its PQ fellowship, grant number 305969/2016-1.

Appendix A. Additional experimental results for missing and full observations

This Appendix contains experiments and evaluation with a set of detailed tables for all 15 domains for missing and full observations. Tables A.3, A.4, A.5, and A.6 detail these experimental results. Each table shows the total number of goal recognition problems used under each domain name (first column). Each row in the tables express averages for the number of candidate goals $|\mathcal{G}|$; the percentage of the plan that is actually observed % Obs; the average number of observations (actions) per problem $|O|$; and for each approach, the time in seconds to recognize the goal given the observations (Time); the accuracy (Acc %) with which the approaches correctly infer the goal; and spread in $\mathcal{G}$ represents the average number of returned goals (S in $\mathcal{G}$). Note that, for every domain, we report the accuracy averaged over all of the problems for each observability. For example, in Table A.4, for the Campus domain, there are 15 problems with 50% observability (totaling 75 for the entire domain), and the IBM 2016 [19] includes this goal in its output for 6 out of 15, resulting in 40% accuracy.

As Tables A.3, A.4, A.5, and A.6 show, our goal recognition heuristics are not only competitive (using thresholds between 10% and 20%) against the other approaches with superior accuracy, but also at least an order of magnitude faster (for all evaluated domains), for example, $\approx$ 2900 times faster than R&G 2010 in DWR domain. Comparison with two other state-of-the-art recent techniques, we can also see that IBM 2016 is substantially slower, even compared to R&G, whereas FGR 2015, while consistently much faster than R&G 2010, is also slower than our heuristics techniques (up to an order of magnitude) across the board. When comparing with our heuristics, the results show that the goal completion heuristic h_{gc} is often more accurate than the uniqueness heuristic h_{uniq}. However, h_{uniq} returns fewer candidate goals (spread in $\mathcal{G}$) than the goal completion heuristic h_{gc} as a result of the *landmark uniqueness value*, which weights landmark information among all landmarks for all goals, making h_{uniq} more precise (but sometimes less accurate) than the goal completion heuristic h_{gc}. We use the threshold value to provide flexibility when the heuristic approaches fail to observe landmarks. While our approach is more accurate than virtually all other approaches with a recognition threshold value of 20% of optimal (sometimes with larger spread), the comparison becomes more complex for other thresholds. The only domain in which the FGR 2015 approach is more accurate than ours is DWR with observability under 70%, however, the spread in $\mathcal{G}$ is nearly twice as large as ours, meaning that FGR 2015 is worse at disambiguating goals. Apart from the Campus and Kitchen domains, our approaches have similar or worse accuracy at very low (30% or less) observability. This loss of accuracy happens for low observability problems because the number of landmarks that happen to be observed is much lower (as the likelihood of observing a landmark goes down) creating a challenge to disambiguate and recognize the correct hidden goal. The results for Campus and Kitchen are explained by the reduced number of goal hypotheses in each domain and the informativeness of the actions, which yield landmarks that favor our approaches. For domains such as DWR, Depots, Sokoban, Zeno-Travel, which are considered more complex because traditional planning heuristics are not very informative for them, our results are mixed. Sometimes, we are able to achieve high accuracy with low observability (albeit with high spread in DWR and Depots), whereas sometimes we achieve lower accuracy with low spread for Sokoban and Zeno-Travel. In this particular setting, Sokoban is known to be a particularly difficult domain for planning heuristics [55], and yields a small number of landmarks per goal. Nevertheless, when our heuristic approaches deal with more than 30% of observability the results are very good both in accuracy and spread in $\mathcal{G}$ for all domains.

Table A.3
Experiments and evaluation with missing and full observations for h_{gc}, h_{uniq}, R&G 2009, and our filtering method (10% of threshold) with R&G 2009 (Part 1).

#	$\lvert\mathcal{G}\rvert$	% Obs	$\lvert O\rvert$	h_{gc}			h_{uniq}			R&G 2009/Filter$_{10\%}$ + R&G 2009		
				Time θ (0/10/20)	Acc % θ (0/10/20)	S in $\mathcal{G}$ θ (0/10/20)	Time θ (0/10/20)	Acc % θ (0/10/20)	S in $\mathcal{G}$ θ (0/10/20)	Time	Acc %	S in $\mathcal{G}$
		10	1.8	0.137/0.143/0.151	39.9%/59.2%/86.4%	1.05/4.62/8.92	0.131/0.139/0.146	31.6%/53.1%/67.1%	1.03/2.77/6.06	1.235/0.631	86.8%/60.9%	7.84/3.34
		30	4.9	0.152/0.160/0.169	50.6%/79.4%/92.1%	1.09/3.96/7.76	0.144/0.153/0.164	51.4%/67.1%/79.4%	1.06/2.51/5.18	1.698/0.819	87.2%/75.7%	3.56/1.79
Blocks-World (1076)	20	50	7.6	0.179/0.187/0.196	65.1%/86.1%/98.7%	1.09/3.13/6.11	0.168/0.174/0.185	60.1%/77.7%/90.1%	1.08/2.24/4.48	2.497/1.008	97.9%/87.2%	2.63/1.42
		70	11.1	0.192/0.201/0.214	84.7%/95.8%/100%	1.12/2.51/3.79	0.184/0.193/0.207	79.1%/90.5%/97.9%	1.13/2.02/3.44	3.704/1.225	97.5%/94.6%	1.83/1.18
		100	14.5	0.245/0.254/0.261	100%/100%/100%	1.09/1.78/2.51	0.238/0.246/0.253	100%/100%/100%	1.09/1.61/2.51	6.123/1.567	100%/100%	1.46/1.06
		10	1	0.031/0.032/0.034	93.3%/100%/100%	1.0/1.33/1.46	0.027/0.029/0.030	100%/100%/100%	1.13/1.46/1.46	0.084/0.082	100%/100%	1.46/1.46
		30	2	0.045/0.048/0.049	100%/100%/100%	1.0/1.33/1.46	0.042/0.044/0.046	100%/100%/100%	1.13/1.46/1.46	0.097/0.096	100%/100%	1.33/1.33
Campus (75)	2	50	3	0.057/0.060/0.064	93.3%/100%/100%	1.0/1.33/1.46	0.055/0.055/0.057	93.3%/100%/100%	1.13/1.46/1.46	0.104/0.102	100%/100%	1.33/1.33
		70	4.4	0.061/0.063/0.068	100%/100%/100%	1.0/1.33/1.33	0.058/0.059/0.062	100%/100%/100%	1.0/1.33/1.33	0.115/0.113	100%/100%	1.26/1.26
		100	5.5	0.066/0.067/0.070	100%/100%/100%	1.0/1.0/1.0	0.061/0.063/0.064	100%/100%/100%	1.0/1.0/1.0	0.128/0.129	100%/100%	1.13/1.13
		10	3.1	0.348/0.364/0.375	35.7%/66.6%/85.7%	1.17/3.42/4.97	0.331/0.342/0.357	32.1%/48.8%/75%	1.09/2.70/3.86	1.485/0.721	77.3%/66.6%	3.98/2.89
		30	8.6	0.372/0.401/0.428	58.3%/82.1%/96.4%	1.05/2.53/3.73	0.356/0.389/0.410	47.6%/71.4%/91.6%	1.07/2.57/2.84	2.307/1.214	77.3%/76.1%	2.39/1.72
Depots (364)	8.5	50	14.1	0.439/0.482/0.514	76.1%/94.1%/97.6%	1.05/1.77/2.53	0.415/0.447/0.470	71.4%/86.9%/96.4%	1.02/2.10/1.75	3.433/1.532	84.5%/89.2%	1.91/1.33
		70	19.7	0.502/0.555/0.593	89.2%/94.1%/97.6%	1.01/1.35/1.91	0.481/0.528/0.562	84.5%/96.4%/96.4%	1.01/1.09/1.46	5.149/1.866	91.6%/94.1%	1.67/1.13
		100	24.4	0.613/0.642/0.677	100%/100%/100%	1.03/1.05/1.65	0.575/0.601/0.633	100%/100%/100%	1.03/1.09/1.46	7.094/2.408	92.8%/100%	1.46/1.07
		10	2.6	0.295/0.302/0.310	41.6%/52.3%/76.1%	1.03/1.88/2.60	0.284/0.292/0.298	35.7%/54.7%/79.7%	1.10/1.84/2.88	1.192/0.499	96.4%/61.9%	4.71/2.40
		30	6.9	0.303/0.310/0.316	54.7%/72.6%/90.1%	1.13/1.90/2.77	0.291/0.299/0.305	47.6%/69.1%/85.7%	1.09/2.03/2.40	1.444/0.605	92.8%/72.6%	3.34/1.66
Driver-Log (364)	10.5	50	11.1	0.312/0.320/0.325	72.6%/85.7%/96.4%	1.16/1.88/2.60	0.290/0.297/0.301	64.2%/82.1%/92.8%	1.14/1.84/2.35	1.608/0.757	94.1%/86.9%	2.88/1.41
		70	15.6	0.322/0.330/0.342	90.4%/94.1%/100%	1.14/1.58/2.15	0.298/0.304/0.309	90.4%/97.6%/100%	1.14/1.47/2.05	1.925/0.846	89.2%/98.8%	2.46/1.35
		100	21.7	0.331/0.339/0.344	100%/100%/100%	1.21/1.32/1.92	0.305/0.311/0.318	100%/100%/100%	1.17/1.25/1.46	2.809/1.213	89.2%/96.4%	2.14/1.14
		10	5.7	0.523/0.534/0.541	36.9%/85.7%/94.1%	1.09/4.32/5.83	0.491/0.499/0.511	33.3%/70.2%/85.7%	1.05/3.07/4.32	1.634/0.921	83.3%/79.7%	4.21/2.29
		30	16	0.535/0.546/0.557	60.7%/95.2%/98.8%	1.03/3.61/4.84	0.518/0.527/0.538	51.1%/80.1%/95.2%	1.05/2.41/3.61	2.977/1.204	80.9%/83.3%	3.34/2.89
DWR (364)	7.25	50	26.2	0.560/0.572/0.581	66.6%/97.6%/100%	1.0/3.19/3.92	0.533/0.541/0.552	61.9%/88.1%/97.6%	1.04/2.16/3.19	4.485/2.121	72.6%/85.7%	2.27/1.64
		70	36.8	0.601/0.608/0.599	89.2%/98.8%/100%	1.0/2.50/3.07	0.540/0.547/0.555	78.5%/98.8%/98.8%	1.03/2/2.50	10.432/3.555	70.2%/89.9%	2.04/1.48
		100	51.9	0.613/0.620/0.626	100%/100%/100%	1.0/1.67/2.50	0.559/0.551/0.564	100%/100%/100%	1.01/1.60/1.67	25.091/5.921	67.8%/92.8%	1.67/1.14
		10	2.9	0.243/0.255/0.262	66.6%/86.2%/94.1%	2.57/3.28/4.41	0.220/0.229/0.237	62.7%/82.3%/92.8%	2.34/3.13/4.09	1.084/0.708	96.1%/85.6%	2.45/2.11
		30	7.8	0.251/0.264/0.273	81.6%/87.5%/88.8%	1.64/2.32/3.28	0.234/0.241/0.251	83.6%/89.5%/90.1%	1.66/2.34/3.28	1.475/0.960	97.3%/87.5%	1.42/1.25
IPC-Grid (673)	9	50	12.7	0.260/0.269/0.276	90.8%/93.4%/93.4%	1.18/1.26/2.32	0.245/0.252/0.259	90.1%/94.7%/94.7%	1.18/1.48/2.57	1.932/1.125	100%/93.4%	1.15/1.04
		70	17.9	0.272/0.278/0.285	97.3%/97.3%/98.1%	1.07/1.15/1.44	0.253/0.260/0.267	97.3%/97.3%/98.1%	1.11/1.16/1.48	2.556/1.211	100%/97.3%	1.05/1.0
		100	24.8	0.286/0.289/0.291	100%/100%/100%	1.0/1.0/1.0	0.261/0.268/0.279	100%/100%/100%	1.0/1.0/1.0	3.868/1.304	100%/100%	1.0/1.0
		10	2.9	0.077/0.083/0.093	58.3%/85.7%/98.8%	1.26/3.19/4.76	0.068/0.083/0.087	58.3%/89.2%/100%	1.17/3.14/3.45	0.511/0.302	98.8%/90.4%	3.36/2.35
		30	7.6	0.084/0.092/0.099	85.7%/97.6%/100%	1.11/2.13/3.25	0.073/0.081/0.088	83.3%/95.2%/100%	1.05/1.90/2.28	0.677/0.399	100%/97.6%	1.76/1.41
Ferry (364)	7.5	50	12.3	0.091/0.096/0.102	95.2%/98.8%/100%	1.07/1.5/1.72	0.084/0.086/0.092	91.6%/92.8%/100%	1.01/1.38/1.40	0.794/0.410	100%/98.8%	1.41/1.16
		70	17.3	0.098/0.100/0.107	100%/100%/100%	1.01/1.13/1.17	0.092/0.094/0.101	100%/100%/100%	1.0/1.11/1.38	1.202/0.525	98.8%/100%	1.14/1.02
		100	24.2	0.104/0.108/0.112	100%/100%/100%	1.0/1.0/1.01	0.099/0.102/0.106	100%/100%/100%	1.0/1.0/1.07	1.693/0.571	100%/100%	1.07/1.0
		10	1.9	0.095/0.098/0.102	62.8%/96.1%/100%	1.14/2.56/5.12	0.077/0.084/0.090	64.7%/100%/100%	1.23/2.54/7.29	0.724/0.444	100%/100%	2.53/2.53
		30	4.5	0.101/0.106/0.108	94.2%/100%/100%	1.01/1.96/2.56	0.083/0.088/0.092	85.7%/100%/100%	1.02/1.96/6.12	0.803/0.486	100%/100%	1.11/1.11
Intrusion (465)	15	50	6.7	0.109/0.111/0.114	99.1%/100%/100%	1.01/1.19/1.61	0.089/0.091/0.094	94.2%/100%/100%	1.04/1.91/3.31	0.888/0.513	100%/100%	1.02/1.0
		70	9.5	0.113/0.115/0.121	100%/100%/100%	1.0/1.02/1.19	0.093/0.096/0.099	94.2%/100%/100%	1.0/1.67/2.46	1.012/0.539	100%/100%	1.0/1.0
		100	13.1	0.120/0.126/0.129	100%/100%/100%	1.0/1.02/1.02	0.098/0.100/0.102	100%/100%/100%	1.0/1.60/1.88	1.257/0.550	100%/100%	1.0/1.0

Table A.4
Experiments and evaluation with missing and full observations for R&G 2010 using Fast-Downward with LM-Cut heuristic, FGR 2015 [18], IBM 2016 using TK* with LM-Cut heuristic (top-1000 plans), and M+L 2018 [24] using Fast-Forward with EHC heuristic (Part 1).

#	$\|\mathcal{G}\|$	% Obs	$\|O\|$	R&G 2010/Filter$_{10\%}$ + R&G 2010 (Fast-Downward with LM-Cut heuristic)			FGR 2015			IBM 2016/Filter$_{10\%}$ + IBM 2016 (TK* with LM-Cut heuristic, top-1000)			M+L 2018 (Fast-Forward with EHC heuristic)		
				Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$
Blocks-World (1076)	20	10	1.8	1271.282/398.090	41.4%/68.2%	11.41/3.34	36.562	65.8%	9.11	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		30	4.9	1280.655/481.905	75.2%/47.1%	7.76/1.84	36.648	78.1%	10.53	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		50	7.6	1284.269/445.196	84.1%/52.4%	7.24/1.78	34.290	81.3%	10.68	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		70	11.1	1296.773/395.902	88.2%/59.3%	8.23/1.67	37.056	89.8%	8.63	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		100	14.5	1305.220/288.751	94.5%/65.2%	8.66/1.71	40.405	100.0%	1.22	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
Campus (75)	2	10	1	1.021/1.038	93.3%/93.3%	1.33/1.33	0.717	53.3%	1.0	45.749/44.834	53.3%/53.3%	1.0/1.0	0.077	86.7%	1.13
		30	2	1.113/1.090	100.0%/100.0%	1.0/1.0	0.696	80.0%	1.13	48.438/48.112	53.3%/53.3%	1.0/1.0	0.076	86.7%	1.27
		50	3	1.333/1.285	100.0%/100.0%	1.0/1.0	0.676	66.6%	1.26	54.111/53.402	40.0%/40.0%	1.0/1.0	0.072	93.3%	1.13
		70	4.4	1.725/1.706	100.0%/100.0%	1.0/1.0	0.668	86.6%	1.6	89.708/89.037	53.3%/53.3%	1.0/1.0	0.071	66.7%	1.0
		100	5.5	1.809/1.771	100.0%/100.0%	1.0/1.0	0.631	93.3%	1.53	183.123/182.910	60.0%/60.0%	1.0/1.0	0.071	60.0%	1.0
Depots (364)	8.5	10	3.1	1347.85/1166.02	59.5%/50.0%	4.10/1.61	†	†	†	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		30	8.6	1369.22/1037.97	44.0%/52.3%	2.44/0.90	†	†	†	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		50	14.1	1335.18/1034.36	48.8%/50.0%	2.86/0.75	†	†	†	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		70	19.7	1392.55/853.40	55.9%/61.9%	3.32/0.74	†	†	†	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		100	24.4	1370.81/670.99	67.8%/75.0%	4.39/0.72	†	†	†	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
Driver-Log (364)	10.5	10	2.6	737.530/509.306	51.2%/52.4%	1.64/0.89	79.487	42.8%	1.91	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		30	6.9	846.176/438.371	50.0%/60.7%	1.58/0.74	60.168	70.2%	3.19	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		50	11.1	851.659/379.450	48.8%/70.2%	1.27/0.75	64.427	79.7%	4.59	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		70	15.6	891.158/308.775	54.8%/91.7%	1.61/0.93	75.084	82.1%	4.10	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		100	21.7	945.013/196.093	46.4%/96.4%	1.39/0.96	96.091	96.4%	1.11	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
DWR (364)	7.25	10	5.7	1246.506/1079.001	61.9%/56.0%	2.99/1.63	66.496	92.8%	6.38	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		30	16	1476.392/1129.304	29.8%/39.3%	1.63/0.75	54.461	97.6%	6.56	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		50	26.2	1501.524/1065.484	33.3%/40.5%	2.01/0.62	56.255	98.8%	6.27	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		70	36.8	1505.305/980.959	34.5%/48.8%	2.17/0.63	65.101	98.8%	6.0	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		100	51.9	1351.309/891.953	57.1%/57.1%	3.57/0.64	86.459	100.0%	1.0	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
IPC-Grid (673)	9	10	2.9	259.349/127.138	66.0%/62.7%	2.46/1.35	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	0.758	58.2%	2.11
		30	7.8	377.482/143.669	85.6%/81.0%	1.44/0.90	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	0.714	80.4%	1.62
		50	12.7	516.035/100.172	85.0%/90.2%	1.39/0.93	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	0.699	93.5%	1.21
		70	17.9	639.157/140.685	81.7%/94.8%	1.76/0.95	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	0.677	91.5%	1.08
		100	24.8	708.007/177.43	93.4%/100.0%	2.54/1.0	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	0.652	83.6%	1.10
Ferry (364)	7.5	10	2.9	251.648/120.596	89.3%/90.5%	2.08/1.44	6.659	91.6%	6.65	*Timeout*	*Timeout*	*Timeout*	4.548	51.2%	1.35
		30	7.6	425.151/91.857	83.3%/97.6%	1.31/1.06	6.801	100.0%	7.57	*Timeout*	*Timeout*	*Timeout*	4.393	73.8%	1.13
		50	12.3	662.567/54.453	66.7%/98.8%	1.24/1.02	8.296	100.0%	7.57	*Timeout*	*Timeout*	*Timeout*	4.310	81.0%	1.12
		70	17.3	820.501/40.898	59.5%/100.0%	1.26/1.0	10.649	100.0%	7.32	*Timeout*	*Timeout*	*Timeout*	4.274	96.4%	1.05
		100	24.2	1015.216/46.148	50.0%/100.0%	1.29/1.0	13.625	100.0%	1.07	*Timeout*	*Timeout*	*Timeout*	4.214	100.0%	1.0
Intrusion (465)	15	10	1.9	5.683/2.889	73.3%/81.0%	3.66/2.37	0.475	89.5%	3.18	*Timeout*	*Timeout*	*Timeout*	0.195	60.0%	1.53
		30	4.5	5.908/4.348	100.0%/100.0%	1.11/1.11	0.476	90.5%	1.88	*Timeout*	*Timeout*	*Timeout*	0.173	93.3%	1.10
		50	6.7	6.248/4.807	100.0%/100.0%	1.02/1.02	0.496	94.3%	1.45	*Timeout*	*Timeout*	*Timeout*	0.162	100.0%	1.04
		70	9.5	6.665/5.261	100.0%/100.0%	1.0/1.0	0.637	99.1%	1.05	*Timeout*	*Timeout*	*Timeout*	0.159	100.0%	1.0
		100	13.1	7.372/5.815	100.0%/100.0%	1.0/1.0	0.828	100.0%	1.04	*Timeout*	*Timeout*	*Timeout*	0.156	100.0%	1.0

Table A.5
Experiments and evaluation with missing and full observations for h_{gc}, h_{uniq}, R&G 2009, and our filtering method (10% of threshold) with R&G 2009 (Part 2).

#	$\|\mathcal{G}\|$	% Obs	$\|O\|$	h_{gc}			h_{uniq}			R&G 2009/Filter$_{10\%}$ + R&G 2009		
				Time θ (0/10/20)	Acc % θ (0/10/20)	S in $\mathcal{G}$ θ (0/10/20)	Time θ (0/10/20)	Acc % θ (0/10/20)	S in $\mathcal{G}$ θ (0/10/20)	Time	Acc %	S in $\mathcal{G}$
Kitchen (75)	3	10	1.3	0.003/0.003/0.004	93.3%/100%/100%	1.46/2.33/3	0.002/0.003/0.003	100%/100%/100%	1.33/2.60/3	0.085/0.084	100%/100%	1.86/1.86
		30	3.5	0.004/0.005/0.006	93.3%/100%/100%	1.46/2.33/3	0.003/0.004/0.005	100%/100%/100%	1.33/2.60/3	0.097/0.098	100%/100%	1.33/1.33
		50	4	0.003/0.003/0.003	93.3%/100%/100%	1.46/2.33/3	0.006/0.005/0.007	100%/100%/100%	1.33/2.60/3	0.104/0.103	100%/100%	1.46/1.46
		70	5	0.008/0.008/0.009	93.3%/100%/100%	1.46/2.60/2.60	0.006/0.007/0.007	100%/100%/100%	1.46/2.33/2.60	0.115/0.116	100%/100%	1.26/1.26
		100	7.4	0.008/0.009/0.009	100%/100%/100%	1.0/1.0/1.0	0.007/0.006/0.008	100%/100%/100%	1.0/1.0/1.0	0.119/0.115	100%/100%	1.26/1.26
Logistics (673)	10.5	10	2.9	0.614/0.618/0.629	55.5%/84.3%/94.7%	1.72/3.58/4.22	0.563/0.572/0.579	55.5%/77.1%/95.4%	1.24/2.84/3.50	1.201/0.705	99.3%/90.8%	2.98/2.69
		30	8.2	0.632/0.637/0.644	80.3%/95.4%/99.3%	1.20/2.14/3.58	0.571/0.580/0.584	76.4%/86.9%/98.1%	1.20/1.96/2.84	1.798/1.166	98.6%/97.3%	1.39/1.35
		50	13.4	0.656/0.662/0.675	90.1%/99.3%/100%	1.10/1.92/2.14	0.599/0.603/0.607	86.2%/95.4%/100%	1.10/1.50/1.77	2.545/1.212	98.6%/98.6%	1.29/1.15
		70	18.9	0.670/0.674/0.683	96.7%/98.6%/100%	1.05/1.39/1.57	0.608/0.611/0.622	96.7%/98.6%/100%	1.05/1.47/1.50	3.460/1.503	100%/100%	1.13/1.11
		100	26.5	0.681/0.685/0.692	100%/100%/100%	1.0/1.0/1.27	0.615/0.624/0.631	100%/100%/100%	1.0/1.0/1.08	4.887/1.691	100%/100%	1.0/1.0
Miconic (364)	6	10	3.9	0.350/0.361/0.366	67.8%/98.8%/100%	1.33/3.28/4.34	0.321/0.325/0.334	54.7%/97.6%/100%	1.26/3.40/4.05	0.838/0.521	100%/98.8%	3.26/3.15
		30	11.1	0.357/0.369/0.370	96.4%/100%/100%	1.10/2.27/3.86	0.326/0.333/0.341	90.1%/100%/100%	1.08/2.53/3.86	1.196/0.707	100%/100%	1.58/1.55
		50	18.1	0.368/0.373/0.375	96.4%/100%/100%	1.01/1.54/2.11	0.339/0.342/0.352	96.4%/100%/100%	1.01/1.47/1.83	1.722/1.099	100%/100%	1.28/1.27
		70	25.3	0.372/0.378/0.384	100%/100%/100%	1.01/1.20/1.57	0.344/0.357/0.365	100%/100%/100%	1.0/1.21/1.57	2.504/1.516	100%/100%	1.03/1.03
		100	35.6	0.389/0.394/0.397	100%/100%/100%	1.0/1.0/1.54	0.356/0.363/0.372	100%/100%/100%	1.0/1.0/1.47	5.105/2.013	100%/100%	1.0/1.0
Rovers (364)	6	10	3	0.342/0.343/0.350	64.2%/91.6%/96.4%	1.72/2.45/3.83	0.310/0.318/0.324	51.1%/79.7%/95.2%	1.10/3.01/3.42	0.704/0.512	98.8%/95.2%	2.85/2.29
		30	7.9	0.347/0.358/0.361	83.3%/91.1%/100%	1.23/2.14/3.72	0.323/0.322/0.335	69.1%/90.1%/97.6%	1.07/2.19/2.46	1.029/0.787	100%/97.6%	1.66/1.41
		50	12.7	0.374/0.383/0.375	92.8%/96.4%/100%	1.08/1.72/3.01	0.331/0.338/0.344	85.7%/95.2%/97.6%	1.01/1.57/2.19	1.355/0.841	100%/98.8%	1.29/1.17
		70	17.9	0.389/0.382/0.391	98.8%/100%/100%	1.01/1.35/2.14	0.345/0.350/0.353	91.6%/98.8%/100%	1.0/1.20/1.58	1.796/1.008	100%/100%	1.07/1.05
		100	24.9	0.392/0.394/0.396	100%/100%/100%	1.0/1.07/1.35	0.356/0.361/0.365	100%/100%/100%	1.0/1.03/1.25	2.292/1.314	100%/100%	1.07/1.0
Satellite (364)	6.5	10	2.1	0.458/0.466/0.471	57.1%/85/7%/100%	1.55/2.33/2.88	0.431/0.445/0.456	47.6%/79.7%/96.4%	1.21/2.09/2.33	1.049/0.599	97.6%/90.4%	3.41/3.02
		30	5.4	0.465/0.474/0.482	76.1%/94.1%/100%	1.31/1.92/2.13	0.442/0.454/0.460	69.1%/89.2%/97.6%	1.14/1.91/2.09	1.182/0.723	97.6%/96.4%	2.40/1.94
		50	8.7	0.472/0.485/0.494	85.7%/98.8%/100%	1.09/1.48/1.91	0.458/0.463/0.477	80.9%/91.6%/98.8%	1.10/1.63/1.91	1.398/0.901	97.6%/97.6%	1.69/1.47
		70	12.2	0.489/0.490/0.498	97.6%/100%/100%	1.07/1.48/1.75	0.460/0.471/0.486	94.1%/98.8%/100%	1.03/1.34/1.63	1.884/1.076	96.4%/100%	1.52/1.25
		100	16.8	0.491/0.499/0.512	100%/100%/100%	1.02/1.21/1.63	0.475/0.482/0.490	100%/100%/100%	1.07/1.21/1.50	2.107/1.224	96.4%/100%	1.33/1.10
Sokoban (364)	7.25	10	3.1	0.549/0.552/0.554	53.5%/86.9%/88.1%	2.05/2.89/3.78	0.523/0.530/0.539	51.1%/67.8%/88.1%	1.85/2.78/3.04	3.025/1.857	69.1%/71.4%	4.02/2.51
		30	8.7	0.555/0.560/0.562	57.1%/77.3%/84.5%	1.36/1.81/2.69	0.531/0.538/0.543	55.9%/69.1%/84.5%	1.21/1.77/2.69	4.429/2.081	89.2%/76.1%	4.10/1.67
		50	14.1	0.568/0.571/0.573	71.4%/88.1%/94.1%	1.32/1.80/2.02	0.540/0.544/0.551	69.1%/83.3%/91.6%	1.20/1.80/1.82	7.553/2.409	89.2%/85.7%	4.16/1.63
		70	19.8	0.577/0.580/0.585	83.3%/91.6%/96.4%	1.04/1.31/1.80	0.554/0.556/0.558	86.9%/92.8%/95.2%	1.08/1.60/1.77	9.112/2.572	89.2%/86.9%	4.17/1.19
		100	35.5	0.586/0.591/0.598	100%/100%/100%	1.0/1.0/1.0	0.562/0.572/0.574	100%/100%/100%	1.0/1.03/1.28	12.008/2.610	89.2%/100%	4.53/1.03
Zeno-Travel (364)	7.5	10	2.6	0.502/0.511/0.528	39.2%/55.9%/80.9%	1.15/1.92/3.04	0.491/0.502/0.509	36.9%/48.8%/70.2%	1.04/1.92/2.13	1.834/1.207	96.4%/66.6%	3.41/1.58
		30	6.7	0.517/0.523/0.536	70.2%/78.5%/91.6%	1.10/1.73/2.26	0.504/0.515/0.520	60.7%/79.7%/90.4%	1.02/1.71/1.73	2.528/1.396	88.1%/79.7%	2.11/1.29
		50	10.8	0.521/0.534/0.544	78.5%/86.9%/95.2%	1.07/1.40/1.71	0.516/0.521/0.528	76.1%/88.1%/95.2%	1.0/1.57/1.61	3.071/1.513	92.8%/90.4%	1.41/1.14
		70	15.2	0.535/0.542/0.550	97.6%/97.6%/100%	1.04/1.14/1.40	0.522/0.533/0.539	90.4%/95.2%/100%	1.0/1.29/1.57	3.986/1.605	96.4%/100%	1.13/1.0
		100	21.1	0.548/0.555/0.564	100%/100%/100%	1.0/1.0/1.07	0.530/0.541/0.552	100%/100%/100%	1.0/1.07/1.10	4.815/1.722	100%/100%	1.07/1.0

Table A.6
Experiments and evaluation with missing and full observations for R&G 2010 using FAST-DOWNWARD with LM-Cut heuristic, FGR 2015 [18], IBM 2016 using TK* with LM-Cut heuristic (top-1000 plans), and M+L 2018 [24] using FAST-FORWARD with EHC heuristic (Part 2).

#	$\|\mathcal{G}\|$	% Obs	$\|O\|$	R&G 2010/Filter$_{10\%}$ + R&G 2010 (FAST-DOWNWARD with LM-Cut heuristic)			FGR 2015			IBM 2016/Filter$_{10\%}$ + IBM 2016 (TK* with LM-Cut heuristic, top-1000)			M+L 2018 (FAST-FORWARD with EHC heuristic)		
				Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$
KITCHEN (75)	3	10	1.3	1.310/1.222	93.3%/93.3%	1.33/1.33	0.373	100.0%	1.86	*Timeout*	*Timeout*	*Timeout*	0.067	86.7%	1.87
		30	3.5	1.365/1.238	93.3%/93.3%	1.20/1.13	0.360	100.0%	1.33	*Timeout*	*Timeout*	*Timeout*	0.064	100.0%	1.80
		50	4	1.571/1.438	100.0%/100.0%	1.33/1.33	0.392	100.0%	1.33	*Timeout*	*Timeout*	*Timeout*	0.060	93.3%	2.24
		70	5	1.702/1.609	100.0%/100.0%	1.20/1.20	0.378	100.0%	1.20	*Timeout*	*Timeout*	*Timeout*	0.057	100.0%	2.20
		100	7.4	2.144/1.983	100.0%/100.0%	1.40/1.40	0.483	100.0%	1.40	*Timeout*	*Timeout*	*Timeout*	0.055	100.0%	1.93
LOGISTICS (673)	10.5	10	2.9	334.315/288.357	65.4%/69.3%	3.57/1.52	†	†	†	*Timeout*	*Timeout*	*Timeout*	3.967	51.0%	1.63
		30	8.2	411.948/321.727	81.7%/83.0%	2.26/0.99	†	†	†	*Timeout*	*Timeout*	*Timeout*	3.830	77.8%	1.46
		50	13.4	431.775/218.848	78.4%/88.9%	1.97/0.93	†	†	†	*Timeout*	*Timeout*	*Timeout*	5.085	88.9%	1.32
		70	18.9	409.629/181.421	83.0%/92.2%	2.22/0.94	†	†	†	*Timeout*	*Timeout*	*Timeout*	3.739	95.4%	1.30
		100	26.5	310.563/109.189	91.8%/95.1%	2.43/0.95	†	†	†	*Timeout*	*Timeout*	*Timeout*	3.213	100.0%	1.10
MICONIC (364)	6	10	3.9	313.117/258.977	89.3%/92.9%	2.07/1.63	†	†	†	*Timeout*	*Timeout*	*Timeout*	9.548	58.3%	1.54
		30	11.1	590.095/338.774	59.5%/88.1%	1.18/1.0	†	†	†	*Timeout*	*Timeout*	*Timeout*	9.286	77.4%	1.26
		50	18.1	578.263/224.679	61.9%/100.0%	1.10/1.04	†	†	†	*Timeout*	*Timeout*	*Timeout*	9.238	88.1%	1.21
		70	25.3	577.087/123.984	61.9%/100.0%	1.07/1.0	†	†	†	*Timeout*	*Timeout*	*Timeout*	9.167	96.4%	1.12
		100	35.6	155.422/30.114	100.0%/100.0%	1.0/1.0	†	†	†	*Timeout*	*Timeout*	*Timeout*	9.036	100.0%	1.0
ROVERS (364)	6	10	3	551.746/524.997	88.1%/48.8%	4.18/0.95	†	†	†	*Timeout*	*Timeout*	*Timeout*	20.893	65.5%	1.85
		30	7.9	589.213/518.466	94.0%/57.1%	3.29/0.64	†	†	†	*Timeout*	*Timeout*	*Timeout*	22.714	81.0%	1.42
		50	12.7	640.802/518.649	88.1%/56.0%	3.20/0.62	†	†	†	*Timeout*	*Timeout*	*Timeout*	17.119	90.5%	1.13
		70	17.9	641.881/517.961	81.0%/57.1%	3.04/0.61	†	†	†	*Timeout*	*Timeout*	*Timeout*	17.095	98.8%	1.07
		100	24.9	589.669/518.319	85.7%/57.1%	3.0/0.57	†	†	†	*Timeout*	*Timeout*	*Timeout*	16.571	100.0%	1.04
SATELLITE (364)	6.5	10	2.1	477.756/418.734	65.5%/81.0%	2.40/1.74	14.821	89.3%	4.86	*Timeout*	*Timeout*	*Timeout*	2.226	69.0%	2.05
		30	5.4	488.884/278.747	79.8%/89.3%	1.98/1.30	32.172	86.9%	4.21	*Timeout*	*Timeout*	*Timeout*	2.024	83.3%	1.42
		50	8.7	553.301/208.331	79.8%/90.5%	1.79/1.11	51.567	88.1%	3.65	*Timeout*	*Timeout*	*Timeout*	1.905	92.9%	1.27
		70	12.2	520.356/186.682	79.8%/94.0%	1.54/1.06	75.363	92.8%	2.89	*Timeout*	*Timeout*	*Timeout*	1.893	98.8%	1.06
		100	16.8	455.197/172.089	82.1%/96.4%	1.71/1.04	113.381	100.0%	2.57	*Timeout*	*Timeout*	*Timeout*	1.786	100.0%	1.04
SOKOBAN (364)	7.25	10	3.1	637.342/377.933	70.8%/70.8%	1.31/0.85	461.701	67.8%	2.98	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		30	8.7	850.272/310.082	51.4%/62.5%	1.15/0.63	370.412	83.3%	3.14	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		50	14.1	1029.601/310.888	38.9%/68.1%	1.19/0.74	358.028	82.1%	2.27	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		70	19.8	1082.685/176.873	37.5%/86.1%	1.11/0.88	353.721	85.7%	1.84	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
		100	35.5	1153.979/108.782	29.2%/95.8%	1.21/0.96	353.183	85.7%	1.03	*Timeout*	*Timeout*	*Timeout*	*Out of Memory*	*Out of Memory*	*Out of Memory*
ZENO-TRAVEL (364)	7.5	10	2.6	782.170/405.126	45.2%/53.6%	1.70/0.76	93.917	66.6%	1.63	*Timeout*	*Timeout*	*Timeout*	13.119	41.7%	1.24
		30	6.7	829.058/458.575	46.4%/66.7%	1.27/0.76	88.285	78.6%	2.27	*Timeout*	*Timeout*	*Timeout*	12.881	76.2%	1.31
		50	10.8	884.339/382.015	39.3%/71.4%	1.13/0.75	105.814	91.6%	2.56	*Timeout*	*Timeout*	*Timeout*	12.513	83.3%	1.07
		70	15.2	922.641/221.105	38.1%/81.0%	1.07/0.81	125.652	94.1%	2.58	*Timeout*	*Timeout*	*Timeout*	12.417	96.4%	1.02
		100	21.1	949.088/153.976	39.3%/89.3%	1.07/0.89	168.674	100.0%	1.0	*Timeout*	*Timeout*	*Timeout*	12.321	100.0%	1.0

Appendix B. Additional experimental results for missing, noisy, and full observations

This Appendix contains experiments and evaluation with a set of detailed tables for all 4 domains for missing, noisy, and full observations. Tables B.7 and B.8 compare results for the experiments with missing, noisy, and full observations for our goal recognition heuristics (using a threshold between 0% and 10%) against R&G 2009 [14], R&G 2010 [15], FGR 2015 [18], IBM 2016 [19], and M+L 2018 [24]. We use our filtering method with 10% of threshold alongside all these three approaches, denoted as Filter$_{10\%}$. For this set of experiments, we used the same 4 domains used by Sohrabi et al. [19], more specifically, Campus, Intrusion, IPC-Grid and Kitchen. In these experiments, column $|N|$ represents the average number of noisy observations, i.e., extra observations that we added randomly to the observation sequence O. These two extra observations represent 12% of noise regarding the total number of observations [19]. Since Sohrabi et al. [19] timed out for all recognition problems, we are unable to provide a direct comparison of accuracy and runtime performance against this approach. However, given our understanding of the underlying technique, we believe that our approaches are almost certainly more computationally efficient, since they use a top-K planner [45] during the recognition process (extracting 1000 sampled plans), much like [15]. The FGR 2015 approach is closer to ours in runtime performance for noisy observations, but still two to ten times slower. Note that the runtime performance of M+L 2018 is quite similar to the performance of our heuristic approaches, but still not as fast as our heuristic approaches.

Under noisy observations, it is clear from the results in Tables B.7 and B.8 (Appendix B) that the approaches R&G 2009 and R&G 2010 are not only much slower but substantially less accurate (with threshold value of 10%) than our heuristics for virtually all 4 domains, reaching a low of 4.4 and 3.3 percent (respectively) of accuracy in the IPC-Grid domain. However, using the recognition threshold = 0%, the R&G 2009 and R&G 2010 approaches are more accurate than our heuristics for two particular domains, more specifically, for Intrusion and Campus (respectively), while the FGR 2015 approach is more accurate than ours for the Intrusion domain, as well as Campus and Kitchen under some conditions (multiple noisy observations). Our uniqueness heuristic h_{uniq} performed better (more accurate and faster) than the goal completion heuristic h_{gc} for all 4 domains. Regarding the difference in accuracy for h_{gc} and h_{uniq} in the Kitchen domain under low observability, note the number of useful actions actually observed (2.5, 2 of which are known to be noise) at that observability level. This means that on average, in this domain, each experiment will have seen mostly noise and possibly one or two actions, or at times, no non-noisy action. Under these conditions being able to get the most information out of the observation (and correctly ignoring noise) is key. Here, the analysis of propositions that are not landmarks performed by the FGR 2015 approach seems to allow coping with a substantial amount of noisy versus non-noisy observations better than our approach. Note that by increasing the threshold parameter, we increase the spread to closer values to FGR 2015 and reach similar levels of accuracy. As can be seen in Table B.8, the results of M+L 2018 under noisy observations are quite good both all evaluated metrics when dealing with more than 25% of observability. M+L 2018 has shown to be both fast and accurate for recognizing goals with noisy observations.

Table B.7
Experiments and evaluation with missing, noisy, and full observations for h_{gc}, h_{uniq}, R&G 2009, and our filtering method (10% of threshold) with R&G 2009 (Part 1).

#	$\|\mathcal{G}\|$	% Obs	$\|N\|$	$\|O+N\|$	h_{gc}			h_{uniq}			R&G 2009/Filter$_{10\%}$ + R&G 2009		
					Time θ(0/10)	Acc % θ(0/10)	S in $\mathcal{G}$ θ (0/10)	Time θ (0/10)	Acc % θ (0/10)	S in $\mathcal{G}$ θ (0/10)	Time	Acc %	S in $\mathcal{G}$
Campus (516)	2	25	2	3.1	0.031/0.034	68.2%/89.9%	1.0/1.28	0.030/0.032	82.1%/90.6%	1.13/1.44	0.073/0.060	88.3%/78.2%	1.27/1.13
		50	2	4.5	0.033/0.035	75.9%/93.7%	1.0/1.20	0.031/0.032	78.2%/93.7%	1.02/1.43	0.076/0.068	89.9%/82.1%	1.26/1.09
		75	2	6.4	0.035/0.039	73.6%/96.2%	1.0/1.22	0.034/0.036	73.6%/96.1%	1.0/1.42	0.079/0.071	90.6%/85.2%	1.27/1.10
		100	2	7.5	0.038/0.041	72.1%/95.3%	1.0/1.23	0.037/0.039	72.1%/95.3%	1.0/1.41	0.084/0.080	89.1%/85.2%	1.22/1.06
Intrusion (300)	16.6	25	2	3.6	0.125/0.127	33.3%/68.8%	1.11/4.43	0.102/0.032	30.0%/58.8%	1.11/3.94	0.537/0.456	71.1%/63.3%	2.65/2.34
		50	2	6.7	0.134/0.135	83.3%/93.3%	1.06/2.04	0.116/0.118	64.4%/88.8%	1.03/2.68	0.649/0.483	95.5%/94.4%	1.28/1.27
		75	2	10.2	0.146/0.150	94.4%/98.8%	1.01/1.33	0.124/0.130	87.7%/94.4%	1.03/1.82	0.712/0.524	100%/98.8%	1.01/1.01
		100	2	15.1	0.155/0.159	100%/100%	1.0/1.10	0.136/0.138	100%/100%	1.0/1.63	0.805/0.659	100%/100.0%	1.0/1.0
IPC-Grid (300)	8.3	25	2	4.1	0.253/0.260	58.8%/75.5%	1.76/2.95	0.208/0.211	53.3%/75.5%	1.72/2.83	0.462/0.301	12.2%/11.1%	7.55/2.81
		50	2	7.6	0.261/0.267	85.5%/85.5%	1.33/1.71	0.212/0.220	83.3%/86.6%	1.33/1.71	0.469/0.312	4.4%/4.4%	8.06/1.61
		75	2	11.5	0.269/0.272	94.4%/94.4%	1.08/1.23	0.224/0.233	94.4%/94.4%	1.08/1.15	0.475/0.323	6.6%/7.7%	7.88/1.10
		100	2	16.9	0.275/0.288	100%/100%	1.0/1.0	0.239/0.246	100%/100%	1.0/1.0	0.476/0.330	10.0%/10.0%	7.76/1.0
Kitchen (150)	3	25	2	2.5	0.094/0.097	11.1%/11.1%	0.22/0.22	0.081/0.083	88.8%/88.8%	2.55/2.55	0.139/0.098	71.1%/62.2%	1.57/1.46
		50	2	4.8	0.095/0.099	28.8%/31.1%	0.64/0.66	0.084/0.088	64.4%/66.6%	1.71/1.73	0.135/0.102	57.7%/51.1%	1.42/1.20
		75	2	7.3	0.097/0.101	24.4%/24.4%	0.66/0.66	0.090/0.092	57.7%/57.7%	1.66/1.66	0.138/0.103	57.7%/48.8%	1.31/1.24
		100	2	11	0.104/0.105	60.0%/60.0%	0.93/0.93	0.093/0.095	66.6%/66.6%	1.13/1.13	0.144/0.109	60.0%/66.6%	1.46/1.13

Table B.8
Experiments and evaluation with missing, noisy, and full observations for R&G 2010 using FAST-DOWNWARD with LM-Cut heuristic, FGR 2015 [18], IBM 2016 using TK* with LM-Cut heuristic (top-1000 plans), and M+L 2018 [24] using FAST-FORWARD with EHC heuristic (Part 2).

#	$\|\mathcal{G}\|$	% Obs	$\|N\|$	$\|O+N\|$	R&G 2010/Filter$_{10\%}$ + R&G 2010 (FAST-DOWNWARD with LM-Cut heuristic)			FGR 2015			IBM 2016			M+L 2018(FAST-FORWARD with EHC heuristic)		
					Time θ (0/10)	Acc % θ (0/10)	S in $\mathcal{G}$ θ (0/10)	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$	Time	Acc %	S in $\mathcal{G}$
CAMPUS (516)	2	25	2	3.1	1.958/1.862	88.3%/88.3%	1.24/1.24	0.713	79.8%	1.33	*Timeout*	*Timeout*	*Timeout*	0.054	59.6%	1.0
		50	2	4.5	2.255/2.220	94.5%/94.5%	1.13/1.13	0.666	90.6%	1.67	*Timeout*	*Timeout*	*Timeout*	0.058	62.0%	1.0
		75	2	6.4	2.784/2.731	99.2%/99.2%	1.11/1.11	0.655	94.6%	1.79	*Timeout*	*Timeout*	*Timeout*	0.063	58.9%	1.0
		100	2	7.5	2.808/2.790	99.2%/99.2%	1.10/1.10	0.644	97.7%	1.81	*Timeout*	*Timeout*	*Timeout*	0.071	49.6%	1.0
INTRUSION (300)	16.6	25	2	3.6	6.216/2.750	35.5%/38.8%	0.78/0.81	0.494	43.3%	2.31	*Timeout*	*Timeout*	*Timeout*	0.131	33.3%	1.34
		50	2	6.7	6.792/1.881	74.4%/78.8%	1.10/0.93	0.511	81.1%	1.78	*Timeout*	*Timeout*	*Timeout*	0.148	80.0%	1.25
		75	2	10.2	8.081/1.686	91.1%/93.3%	0.94/0.94	0.654	93.3%	1.10	*Timeout*	*Timeout*	*Timeout*	0.159	96.6%	1.02
		100	2	15.1	8.753/1.670	100%/100%	1.0/1.10	0.885	100.0%	1.06	*Timeout*	*Timeout*	*Timeout*	0.177	100.0%	1.0
IPC-GRID (300)	8.3	25	2	4.1	36.275/20.612	8.8%/10.0%	0.91/1.0	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	0.376	64.4%	1.88
		50	2	7.6	16.310/4.304	3.3%/3.3%	0.98/0.98	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	0.389	83.3%	1.35
		75	2	11.5	33.358/2.737	7.7%/7.7%	1.03/0.92	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	0.412	91.1%	1.16
		100	2	16.9	35.850/4.328	10.0%/10.0%	1.0/1.0	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	*Timeout*	0.525	76.6%	1.10
KITCHEN (150)	3	25	2	2.5	3.038/2.343	53.3%/57.4%	1.35/0.70	0.381	53.3%	1.33	*Timeout*	*Timeout*	*Timeout*	0.017	57.7%	1.55
		50	2	4.8	13.291/5.009	48.8%/44.4%	1.17/0.48	0.410	51.1%	1.22	*Timeout*	*Timeout*	*Timeout*	0.022	62.2%	1.57
		75	2	7.3	6.467/2.756	51.1%/44.4%	1.22/0.53	0.426	53.3%	1.20	*Timeout*	*Timeout*	*Timeout*	0.038	71.1%	1.86
		100	2	11	5.289/1.818	73.3%/66.6%	1.4/0.66	0.538	73.3%	1.26	*Timeout*	*Timeout*	*Timeout*	0.045	66.6%	1.13

References

[1] G. Sukthankar, R.P. Goldman, C. Geib, D.V. Pynadath, H.H. Bui Plan, Activity, and Intent Recognition: Theory and Practice, Elsevier, 2014.
[2] D. Avrahami-Zilberbrand, G.A. Kaminka, Fast and complete symbolic plan recognition, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2005, pp. 653–658.
[3] C.W. Geib, R.P. Goldman, A probabilistic plan recognition algorithm based on plan tree grammars, Artif. Intell. 173 (11) (2009) 1101–1132.
[4] O. Amir, Y.K. Gal, Plan recognition and visualization in exploratory learning environments, ACM Trans. Interact. Intell. Syst. 3 (3) (2013) 16:1–16:23.
[5] R. Mirsky, R. Stern, Y. Gal, M. Kalech, Plan recognition design, in: Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI), 2017, pp. 4971–4972.
[6] C.W. Geib, R.P. Goldman, Plan recognition in intrusion detection systems, in: DARPA Information Survivability Conference and Exposition (DISCEX), 2001.
[7] C.W. Geib, Problems with intent recognition for elder care, in: Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI), 2002, pp. 13–17.
[8] O. Uzan, R. Dekel, O. Seri, Y.K. Gal, Plan recognition for exploratory learning environments using interleaved temporal search, AI Mag. 36 (2) (2015) 10–21.
[9] R. Mirsky, Y.K. Gal, S.M. Shieber, CRADLE: an online plan recognition algorithm for exploratory domains, ACM Trans. Intell. Syst. Technol. 8 (3) (2017) 45:1–45:22.
[10] D.V. Pynadath, M.P. Wellman, Accounting for context in plan recognition, with application to traffic monitoring, Computing Research Repository (CoRR), arXiv:1302.4980 [abs].
[11] R. Granada, R.F. Pereira, J. Monteiro, R. Barros, D. Ruiz, F. Meneguzzi, Hybrid activity and plan recognition for video streams, in: The AAAI 2017 Workshop on Plan, Activity, and Intent Recognition, 2017.
[12] R. Mirsky, Y.K. Gal, D. Tolpin, Session analysis using plan recognition, in: The ICAPS 2017 Workshop on User Interfaces and Scheduling and Planning (UISP@ICAPS), 2017.
[13] L. Amado, R.F. Pereira, J.P. Aires, M. Magnaguagno, R. Granada, F. Meneguzzi, Goal recognition in latent space, in: Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), 2018.
[14] M. Ramírez, H. Geffner, Plan recognition as planning, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2009.
[15] M. Ramírez, H. Geffner, Probabilistic plan recognition using off-the-shelf classical planners, in: Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI), 2010.
[16] D. Pattison, D. Long, Domain independent goal recognition, in: Starting AI Researcher Symposium (STAIRS), 2010.
[17] S. Keren, A. Gal, E. Karpas, Goal recognition design, in: Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2014.
[18] Y. E-Martín, M.D. R.-Moreno, D.E. Smith, A fast goal recognition technique based on interaction estimates, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2015, pp. 761–768.
[19] S. Sohrabi, A.V. Riabov, O. Udrea, Plan recognition as planning revisited, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2016.
[20] R.F. Pereira, F. Meneguzzi, Landmark-based plan recognition, in: Proceedings of the European Conference on Artificial Intelligence (ECAI), 2016.
[21] R.F. Pereira, N. Oren, F. Meneguzzi, Landmark-based heuristics for goal recognition, in: Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI), 2017.
[22] P. Masters, S. Sardiña, Cost-based goal recognition for path-planning, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2017, pp. 750–758.
[23] J. Hoffmann, J. Porteous, L. Sebastia, Ordered landmarks in planning, J. Artif. Intell. Res. 22 (1) (2004) 215–278.
[24] M. Vered, R.F. Pereira, M. Magnaguagno, F. Meneguzzi, G.A. Kaminka, Towards online goal recognition combining goal mirroring and landmarks, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2018.
[25] M. Vered, R.F. Pereira, M.C. Magnaguagno, F. Meneguzzi, G.A. Kaminka, Online goal recognition as reasoning over landmarks, in: The AAAI 2018 Workshop on Plan, Activity, and Intent Recognition, 2018.
[26] S.V. Albrecht, P. Stone, Autonomous agents modelling other agents: a comprehensive survey and open problems, Artif. Intell. 258 (2018) 66–95.
[27] M. Ghallab, D.S. Nau, P. Traverso, Automated Planning and Acting, Elsevier, 2016.
[28] R.E. Fikes, N.J. Nilsson, STRIPS: a new approach to the application of theorem proving to problem solving, J. Artif. Intell. Res. 2 (3) (1971) 189–208.
[29] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, D. Wilkins, PDDL – the Planning Domain Definition Language, in: The Fourth International Conference on Artificial Intelligence Planning Systems (AIPS'98), 1998.
[30] A. Kulkarni, S. Srivastava, S. Kambhampati, A unified framework for planning in adversarial and cooperative environments, in: Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI), 2019.
[31] R.F. Pereira, N. Oren, F. Meneguzzi, Monitoring plan optimality using landmarks and domain-independent heuristics, in: The AAAI 2017 Workshop on Plan, Activity, and Intent Recognition, 2017.
[32] R.F. Pereira, N. Oren, F. Meneguzzi, Detecting commitment abandonment by monitoring sub-optimal steps during plan execution, in: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 2017.
[33] S. Richter, M. Helmert, M. Westphal, Landmarks revisited, in: Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI), 2008.
[34] V. Vidal, H. Geffner, Solving simple planning problems with more inference and no search, in: P. van Beek (Ed.), Principles and Practice of Constraint Programming – CP 2005, Springer, Berlin Heidelberg, 2005.
[35] S. Richter, M. Westphal, The LAMA planner: guiding cost-based anytime planning with landmarks, J. Artif. Intell. Res. 39 (1) (2010) 127–177.
[36] T. Bylander, The computational complexity of propositional STRIPS planning, J. Artif. Intell. Res. 69 (1994) 165–204.
[37] L. Zhu, R. Givan, Landmark extraction via planning graph propagation, in: Printed Notes of ICAPS'03 Doctoral Consortium, June 2003.
[38] E. Keyder, S. Richter, M. Helmert, Sound and complete landmarks for and/or graphs, in: Proceedings of the 19th European Conference on Artificial Intelligence (ECAI), 2010.
[39] J. Hoffmann, B. Nebel, The FF planning system: fast plan generation through heuristic search, J. Artif. Intell. Res. 14 (2001) 253–302.
[40] A.L. Blum, M.L. Furst, Fast planning through planning graph analysis, J. Artif. Intell. Res. 90 (1–2) (1997) 281–300.
[41] M. Helmert, The fast downward planning system, Computing Research Repository (CoRR), arXiv:1109.6051 [abs].
[42] R.F. Pereira, F. Meneguzzi, Goal and Plan Recognition Datasets using Classical Planning Domains, at the data repository Zenodo, Jul. 2017.
[43] E. Keyder, H. Geffner, Heuristics for planning with action costs revisited, in: Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2008), 2008.
[44] M. Helmert, C. Domshlak Landmarks, Critical paths and abstractions: what's the difference anyway?, in: Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2009.
[45] M. Katz, S. Sohrabi, O. Udrea, D. Winterer, A novel iterative approach to top-k planning, in: Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2018.

[46] J. Hong, Goal recognition through goal graph analysis, J. Artif. Intell. Res. 15 (2001) 1–30.
[47] M. Vered, G.A. Kaminka, S. Biham, Online goal recognition through mirroring: humans and agents, in: Proceedings of the Annual Conference on Advances in Cognitive Systems, 2016.
[48] M. Vered, G.A. Kaminka, Heuristic online goal recognition in continuous domains, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2017.
[49] G.A. Kaminka, M. Vered, N. Agmon, Plan recognition in continuous domains, in: Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI), 2018.
[50] S. Keren, A. Gal, E. Karpas, Goal recognition design for non-optimal agents, in: Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI), 2015, pp. 3298–3304.
[51] S. Keren, A. Gal, E. Karpas, Goal recognition design with non-observable actions, in: Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI), 2016, pp. 3152–3158.
[52] R.G. Freedman, Y.R. Fung, R. Ganchin, S. Zilberstein, Towards quicker probabilistic recognition with multiple goal heuristic search, in: The AAAI 2018 Workshop on Plan, Activity, and Intent Recognition, 2018.
[53] D. Davidov, S. Markovitch, Multiple-goal heuristic search, J. Artif. Intell. Res. 26 (2006) 417–451.
[54] N. Lipovetzky, C. Muise, H. Geffner, Traps, invariants, and dead-ends, in: Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2016.
[55] A.G. Pereira, M. Ritt, L.S. Buriol, Optimal Sokoban solving using pattern databases with specific domain knowledge, Artif. Intell. 227 (2015) 52–70.