ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240999

Fact Probability Vector Based Goal Recognition

Nils Wilken[a,*], **Lea Cohausz**[b], **Christian Bartelt**[a] **and Heiner Stuckenschmidt**[b]

[a]Institute for Enterprise Systems, University of Mannheim
[b]Data and Web Science Group, University of Mannheim
ORCID (Nils Wilken): https://orcid.org/0000-0003-1336-245X

Abstract. We present a new approach to goal recognition that involves comparing observed facts with their expected probabilities. These probabilities depend on a specified goal g and initial state s_0. Our method maps these probabilities and observed facts into a real vector space to compute heuristic values for potential goals. These heuristic values estimate the likelihood of a given goal being the true objective of the observed agent. As obtaining exact expected probabilities for observed facts in an observation sequence is often practically infeasible, we propose and empirically validate a method for approximating these probabilities. Our empirical results show that the proposed approach offers improved goal recognition precision compared to state-of-the-art techniques while reducing computational complexity.

1 Introduction

Goal recognition is the task of recognizing the goal(s) of an observed agent given a sequence of actions executed or a sequence of states visited by the agent. In this paper, we focus on observed action sequences for the theoretical discussion of the goal recognition problem. Goal recognition is relevant in many application domains like crime detection [7], pervasive computing [21, 6], or traffic monitoring [13]. Existing goal recognition systems often rely on the principle of Plan Recognition As Planning (PRAP) and, hence, utilize concepts and algorithms from the classical planning community to solve the goal recognition problem [14, 15, 18, 1]. Nevertheless, a fundamental limitation of many systems from this area, which require computing entire plans to solve a goal recognition problem, is their computational complexity. Pereira et al. [12] approached this shortcoming by introducing a planning landmark-based heuristic approach, which outperforms existing goal recognition approaches and requires much less computation time. However, as this approach only relies on fact landmarks, it neglects the information from all other observed facts.

In the context of classical planning, facts model the properties of the planning environment. The subset of true facts in a particular moment defines a planning state. This paper presents a novel approach to goal recognition that involves comparing observed facts with their probability of being observed. We define the probability of observing a fact f as the probability of f being added to the planning state by an action in an observation sequence (i.e., plan) that starts at the initial state s_0 and ends at a possible goal state s_g. The proposed approach implements this comparison by mapping the fact observation probabilities per goal, the initial state s_0, and the currently observed state s_t into a real vector space. Based on these vector mappings, the proposed method computes a heuristic value for each possible goal. These heuristic values estimate the probabilities that a possible goal g is the actual goal g^* of the observed agent. In the remainder of this paper, we refer to our proposed method as Fact Probability Vector Based Goal Recognition (FPV).

FPV and the planning landmark-based (PLR) method introduced by Pereira et al. [12] fall in the same class of heuristic goal recognition approaches. PLR uses the fact that a planning landmark l, by definition, has to be observed during an observation sequence that starts at s_0 and ends at g. Based on this, PLR uses a heuristic that estimates the probability of a goal g being the actual goal g^* by counting how many planning landmarks out of all landmarks have already been observed for g. However, this limits the PLR approach to using only observed facts that are planning landmarks. This problem becomes more severe when fewer planning landmarks exist in a planning domain. One extreme case for such a domain is the grid-world domain, as discussed in Section 3. Only the initial state and the goal facts are fact landmarks in this domain. To address this problem, FPV considers not solely the facts that are landmarks but *all* facts that a given planning domain defines.

More explicitly, the contributions of this paper are:

- In Section 3, we propose a novel approach to goal recognition that involves comparing fact observation probabilities with actually observed facts.
- As the fact observation probabilities are usually not given, we propose a method to estimate the fact observation probabilities in Section 4.
- In Section 5, we empirically show that FPV achieves better goal recognition precision, especially when dealing with low observability, and is more efficient regarding computation time than existing state-of-the-art methods.

2 Background

This section introduces relevant definitions in the context of classical planning and goal recognition.

2.1 Classical Planning

Classical planning uses a symbolic model of the planning domain that defines the properties of the planning environment (i.e., planning facts) and possible actions. These actions are defined by their preconditions and effects. Given an initial state and a goal, planning methods aim to construct an action sequence (i.e., plan) that transforms the initial state into a valid goal state (i.e., a state in which

* Corresponding Author. Email: nils.wilken@uni-mannheim.de.

the goal description holds). The generated plans might be strictly or approximately optimal depending on the planning method used. For the theoretical analysis of FPV, we focus on the STRIPS part of the Planning Domain Definition Language (PDDL) [11]. Nevertheless, our empirical evaluation results show that FPV performs very well in domains not restricted to STRIPS. It is important to note that FPV must properly handle negated facts in the preconditions and goal(s) to apply the STRIPS principle to non-STRIPS domains. Our current implementation compiles negated facts away and introduces an additional fact for each negated fact in the grounded planning domain.

Definition 1 ((STRIPS) Planning Problem)**.** *A Planning Problem is a Tuple $P = \langle F, s_0, A, g\rangle$ where F is a set of facts, $s_0 \subseteq F$ and $g \subseteq F$ are the initial state and the goal and A is a set of actions. Each action is defined by its preconditions $Pre(a) \subseteq F$ and its effects $Add(a) \subseteq F$ and $Del(a) \subseteq F$. $Add(a)$ and $Del(a)$ describe the effects of an action a in terms of facts that are added and deleted from the current state when the planning agent executes a. Actions have a non-negative cost $c(a)$. All facts $f \in F$ that are true in the current planning state define a state $s \subseteq F$. A state s is a goal state if and only if $s \supseteq g$. An action a is applicable in a state s if and only if $Pre(a) \subseteq s$. Applying an action a in a state s leads to a new state $s' = (s \cup Add(a) \setminus Del(a))$.*

Definition 2. *(Solution to a Planning Problem) A solution for a planning problem is a sequence of applicable actions $\pi = (a_i)_{i\in[1,N]}$ that transforms s_0 into a goal state. The cost of a plan is defined as $c(\pi) = \sum_i c(a_i)$. A plan is optimal if the cost of the plan is minimal.*

Ignore Delete Effects Relaxation. In this paper, we propose to use concepts developed in the context of the *Ignore Delete Effects Relaxation*, which has been popular in classical planning since its introduction by Bonet et al. [3]. In the remainder of the paper, we use the term *relaxed planning state* to refer to a delete relaxed planning state. As the name indicates, this relaxation ignores all delete effects of the actions defined in a planning domain. More formally, a delete relaxed STRIPS planning problem is defined as follows:

Definition 3 (Delete Relaxed (STRIPS) Planning Problem)**.** *Given a planning problem $P = \langle F, s_0, A, g\rangle$, the corresponding delete relaxed planning problem is defined as $P^+ = \langle F, s_0, A^+, g\rangle$. The delete relaxed action set A^+ contains all delete relaxed actions from A. The preconditions, add list, and delete list for the delete relaxed version a^+ of action a are defined as $pre_{a^+} = pre_a$, $add_{a^+} = add_a$, $del_{a^+} = \emptyset$. Hence, applying an action a^+ in a state s leads to a new state $s^+ = (s \cup Add(a^+))$.*

Definition 4. *(Solution to a Delete Relaxed Planning Problem) A solution for a delete relaxed planning problem is a sequence of applicable actions (relaxed plan) $\pi^+ = (a_i^+)_{i\in[1,N]}$ that transforms s_0 into a goal state. The cost of a relaxed plan is defined as $c(\pi^+) = \sum_i c(a_i^+)$. A relaxed plan is optimal if the cost of the relaxed plan is minimal.*

2.2 Goal Recognition

This paper investigates a solution method for the (online) goal recognition problem. Let us first define the goal recognition problem:

Definition 5 (Goal Recognition)**.** *Goal recognition is the problem of inferring a nonempty subset $\widehat{G}$ of a set of intended goals G of an observed agent, given a possibly incomplete sequence of observed actions O and a domain model D that describes the environment in which the observed agent acts. The observation sequence O is a plan from s_0 to the agent's hidden true goal g^*. More formally, a goal recognition problem is a tuple $R = \langle D, O, G\rangle$.*

Definition 6. *(Solution to a Goal Recognition Problem) A solution to a goal recognition problem R is a nonempty subset $\widehat{G} \subseteq G$ such that all $g \in \widehat{G}$ are considered to be equally most likely to be the true hidden goal g_* that the observed agent currently tries to achieve.*

The most favorable solution to a goal recognition problem R is a subset $\widehat{G}$ containing only the true hidden goal g_*. In this paper, $D = \langle F, s_0, A\rangle$ is a planning domain with a set of facts F, the initial state s_0, and a set of actions A. The online goal recognition problem is an extension to the previously defined goal recognition problem that additionally introduces the concept of time, and we define it as follows:

Definition 7 (Online Goal Recognition)**.** *Online goal recognition is a variant of the goal recognition problem, where we assume that the observation sequence O is revealed incrementally. More explicitly, let $t \in [1, T]$ be a time index, where $T = |O|$ and hence, the observation sequence for time index t is defined as $O_t = (o_i)_{i\in[1,t]}$. For every value of t, a goal recognition problem $R(t)$ can be induced as $R(t) = \langle D, G, O_t\rangle$.*

Definition 8. *(Solution to an Online Goal Recognition Problem) A solution to the online goal recognition problem are the nonempty subsets $\widehat{G}_t \subseteq G; \forall t \in [1, T]$.*

It is important to note that, in contrast to many existing methods (i.e., [14], [15], [10], [19], [18]), FPV can deal with observing actions *and* observing states simultaneously. FPV requires fact observations, as FPV compares the observed facts with the fact observation probabilities. From action observations, FPV can derive the corresponding observed facts from the given planning domain, which defines the add effects of each action. In the case of state observations, the observed states directly define the observed facts.

3 Fact Probability Vector Based Online Goal Recognition

In this paper, we propose to perform goal recognition by comparing the fact observation probability and the set of actually observed planning facts. To discuss how the fact observation probability is formally defined, we first define the case in which FPV considers a planning fact as being observed.

Definition 9 (Observed Planning Fact)**.** *Given a goal recognition problem R, we define a planning fact $f \in F$ to be observed during an observation sequence O if and only if $f \in \bigcup_{a\in O} add(a)$.*

Definition 10 (Fact Observation Probability)**.** *We define the set of fact observation probabilities for each goal $g \in G$ as $\mathcal{P}_F^g = \{P_f(B|s_0, g)|f \in F\}$, where variable B can take two values; one representing that f is observed (cf., Definition 9) and the other representing that f is not observed.*

For example, the distribution $P_f(B|s_0, g)$ models the probability that fact f is observed during an observation sequence that starts in s_0 and ends in a goal state s_g. FPV implements the comparison between fact observation probabilities and observed facts based on real vector interpretations of $\mathcal{P}_F^g$, s_0, and the currently observed state s_t.

Mapping Planning States And $\mathcal{P}_F^g$ Into a Real Vector Space. To map a planning state s into its $|F|$-dimensional real vector representation $\mathbf{s} \in \mathbb{R}^{|F|}$, we use the following mapping:

$$s_f = \begin{cases} 1, & iff \quad f \in s \\ 0, & else \end{cases} \tag{1}$$

In Equation 1, s is a planning state, and $\mathbf{s}_f$ is the element of $\mathbf{s}$ that encodes the planning fact f. The resulting vector mapping $\mathbf{s}$ of a planning state s encodes the observational evidence from s. Algorithm 1, presented later in the paper, will refer to this function using the signature $mapState(s)$.

To map the fact observation probabilities $\mathcal{P}_F^g$ (cf., Definition 10 into an $|F|$ dimensional real vector representation $\mathbf{v}^g \in \mathbb{R}^{|F|}$, given an initial state s_0 and a goal description g, FPV uses the following mapping:

$$\mathbf{v}_f^g = P_f(f \in \bigcup_{a \in O} add(a) | s_0, g) \tag{2}$$

Algorithm 1, which is presented later in the paper, will refer to this function using the signature $mapP(s_0, g, \mathcal{P}_F^g)$. As an example, consider a simple domain in which $F = \{f_1, f_2, f_3\}$, $s_0 = \emptyset$, and there is only one possible goal $g = \{f_3\}$. Further, $\mathcal{P}_F^g$ is defined as $\mathcal{P}_F^g = \{P_{f_1} = (0.8, 0.2), P_{f_2} = (0.3, 0.7), P_{f_3} = (1.0, 0.0)\}$, where we use the following notation $(P_f(f \in \bigcup_{a \in O} add(a) | s_0, g), P_f(f \notin \bigcup_{a \in O} add(a) | s_0, g))$. For this example, $\mathbf{v}^g$ is defined as $\mathbf{v}^g = (0.8, 0.3, 1.0)$. The resulting vector representation of $\mathcal{P}_F^g$ is an encoding of how probable it is that each fact $f \in F$ is added to the current planning state s_t at any point in time by an observation sequence that starts at s_0 and leads to goal g.

Method. For goal recognition, FPV calculates a heuristic score for each goal based on the mappings of the points $\mathbf{s_0^+}$ and $\mathbf{s_t^+}$ (cf., Equation 1) and the mapping of $\mathcal{P}_F^g$ for each goal (cf., Equation 2). FPV uses the currently observed relaxed planning state (i.e., s_t^+) instead of the non-relaxed state because it contains additional information about the planning states the observation sequence has already visited. As defined by Definition 3, under the delete relaxation, facts can only be added to the initial planning state but can never be deleted. Consequently, s_t^+ contains all facts added by any action in the relaxed observation sequence O^+, which contains the relaxed action a^+ for all actions $a \in O$. The resulting vector $\mathbf{s_t^+}$ contains a one for all facts that *either* have been already true in s_0 *or* were added by an action in the observation sequence and zero otherwise. Based on the vector mappings, FPV calculates each goal's heuristic score as defined by Equation 3.

$$h(\mathbf{s_0^+}, \mathbf{s_t^+}, \mathbf{v}^g) = ||\overrightarrow{(\mathbf{s_0^+} \odot \mathbf{v}^g, \mathbf{v}^g)}||_2 - ||\overrightarrow{(\mathbf{s_t^+} \odot \mathbf{v}^g, \mathbf{v}^g)}||_2 \tag{3}$$

In Equation 3, $\odot$ is an elementwise multiplication of two vectors as defined by Equation 4, $\overrightarrow{(\mathbf{x}, \mathbf{y})}$ is a direction vector between two points $\mathbf{x}$ and $\mathbf{y}$ as defined by Equation 5, and $||\mathbf{x}||_2$ is the l^2 vector norm, which is defined as $||\mathbf{x}||_2 = \sqrt{\sum_{k=1}^{n} x_k^2}$.

$$(\mathbf{s} \odot \mathbf{v})^i = \begin{cases} \mathbf{s}^i * \mathbf{v}^i, & iff \quad \mathbf{v}^i > 0 \\ \mathbf{s}^i, & else \end{cases} \tag{4}$$

Using the elementwise multiplication $\odot$, as defined by Equation 4, ensures that the heuristic value monotonically increases for a goal g when only facts for which $P_f(f \in \bigcup_{a \in O} add(a) | s_0, g) > 0$ holds are observed.

$$\overrightarrow{(\mathbf{x}, \mathbf{y})} = \mathbf{y} - \mathbf{x} \tag{5}$$

Algorithm 1 Fact Observation Probability Based Goal Recognition.

```
1:  function RECOGNIZE GOALS(P_F, s_0, G, O_t)
2:      Ĝ ← ∅                                  ▷ Set of recognized goals
3:      h ← {}                                 ▷ Maps goals to heuristic values
4:      for all g ∈ G do
5:          v^g ← mapP(s_0, g, P_F^g)          ▷ cf., Equation 2
6:      end for
7:      s_t^+ ← s_0^+                          ▷ Initialize current observed state
8:      for all o_i ∈ O_t do
9:          s_t^+ ← s_t^+[[o_i]]               ▷ Update observed state
10:     end for
11:     s_t^+ ← mapState(s_t^+)                ▷ cf., Equation 1
12:     for all g ∈ G do
13:         h[g] ← h(s_0^+, s_t^+, v^g)        ▷ cf., Equation 3
14:     end for
15:     maxH ← maxValue(h)
16:     for all g ∈ G do
17:         if h[g] = maxH then
18:             Ĝ ← Ĝ ∪ {g}
19:         end if
20:     end for
21:     return Ĝ
22: end function
```

Computing the direction between $(\mathbf{s_0^+} \odot \mathbf{v}^g)$ and $\mathbf{v}^g$ results in a vector that encodes for each fact f, how likely it is that f has to be added to the initial state to reach a valid goal state as approximated by $\mathcal{P}_F^g$. This is because the resulting vector becomes zero at all points where *both* $\mathbf{s_0^+}$ and $\mathbf{v}^g$ have values *greater than* zero. Similarly to the first direction vector, computing the direction between $(\mathbf{s_t^+} \odot \mathbf{v}^g)$ and $\mathbf{v}^g$ results in a vector that encodes for each fact f, how likely it is that f has to be added to the currently observed relaxed state $\mathbf{s_t^+}$ to reach a valid goal state. Hence, the heuristic computed by FPV estimates the probability of a goal g being the true hidden goal g^* of the observed agent by computing the distance already covered by the observation sequence between the initial state and a valid goal state for g as estimated by the fact observation probabilities. In addition, the heuristic punishes goals for which $\mathcal{P}_F^g$ assigns a *probability of zero* to facts that were *actually observed*. In this case, the vector resulting from $\overrightarrow{(\mathbf{s_t^+} \odot \mathbf{v}^g, \mathbf{v}^g)}$ will have a value of minus one at all positions that correspond to such facts. Consequently, $||\overrightarrow{(\mathbf{s_t^+} \odot \mathbf{v}^g, \mathbf{v}^g)}||_2$ increases, which results in an overall decreased heuristic value.

The algorithm used to recognize goals based on the previously defined vector mappings is described in Algorithm 1. As a first step, $\mathbf{v}^g$ has to be determined for each goal $g \in G$ (lines 4-6). As a second step, the vector representation $\mathbf{s_t^+}$ of the currently observed relaxed state is calculated (lines 7-9). Afterward, Algorithm 1 calculates the heuristic value for all goals (lines 10-12). As a last step, from the calculated heuristic values, Algorithm 1 selects the goals that have the maximum heuristic value (lines 13-19).

Example. This paragraph illustrates FPV based on a simple example. For this example, we consider the grid environment in Figure 1. In 1, the agent is initially located in cell c23 and wants to reach one of the two goals, $G1$ or $G2$. The agent can move through the environment using moves in all directions *except diagonal moves* to all cells that are not colored black; each move has a cost of 1. We assume that Figure 1 visualizes a corresponding planning domain model. This domain model includes only one predicate (is-at ?x) that models the current position of the agent (e.g., (is-at c23)) and only one action

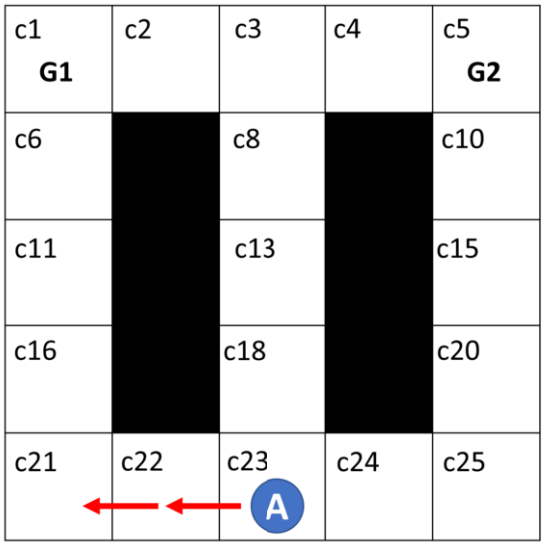

Figure 1: Exemplary grid environment.

schema m(?x, ?y) that allows the agent to move from cell x to cell y. Further, the red arrows indicate actions in the observation sequence O. In addition, we assume that the agent acts entirely rationally according to the modeled action costs (i.e., the agent uses cost-optimal plans).

In this example, both goals have exactly two optimal plans. Further, we assume that the agent is indifferent about taking any of the two optimal plans and, hence, takes either path with a probability of 0.5. Table 1 summarizes the fact observation probabilities for the described example domain. Consequently, for this example, $\mathbf{v}^{G1}$ equals the observed column for P_f^{G1} from top to bottom and $\mathbf{v}^{G2}$ the observed column for P_f^{G2} from top to bottom. The two red arrows in Figure 1 define the observation sequence for this scenario as $O = \{m(c23, c22), m(c22, c21)\}$. Hence, the currently observed relaxed state is $s_t^+ = \{$(is-at c23), (is-at c22), (is-at c21)$\}$. For goal recognition, FPV calculates the heuristic values for G1 and G2. We will exemplify the calculation for G1 only here, as the calculation follows similar paths for G2. As a first step $||\overrightarrow{(\mathbf{s_0^+} \odot \mathbf{v}^{G1}, \mathbf{v}^{G1})}||_2$ is calculated. In this example, as only one fact (i.e., (is-at c23)) is true in s_0, $\mathbf{s_0^+}$ has only one non-zero element representing the fact (is-at c23). Hence, the result of $\mathbf{s_0^+} \odot \mathbf{v}^{G1}$ is similar to $\mathbf{s_0^+}$ as the fact observation probability for (is-at c23) is equal to 1 and all other elements of $\mathbf{v}^{G1}$ are multiplied out. From this, FPV calculates $\overrightarrow{(\mathbf{s_0^+} \odot \mathbf{v}^{G1}, \mathbf{v}^{G1})}$. The resulting vector is, in this example, equal to $\mathbf{v}^{G1}$ except for having a value of 0 at the position that represents the fact (is-at c23) as this has a value of 1 in both vectors. Then, FPV calculates the l^2 norm for this vector, which is 1.87 for this example. As a second step $||\overrightarrow{(\mathbf{s_t^+} \odot \mathbf{v}^{G1}, \mathbf{v}^{G1})}||_2$ is calculated. In contrast to $\mathbf{s_0^+}$, $\mathbf{s_t^+}$ has three non-zero elements representing the facts in s_t^+ (i.e., (is-at c23), (is-at c22), and (is-at c21)). Consequently, the result of $\mathbf{s_t^+} \odot \mathbf{v}^{G1}$ also has three non-zero elements containing the observation probabilities for the facts (is-at c23), (is-at c22), and (is-at c21). Hence, $\overrightarrow{(\mathbf{s_t^+} \odot \mathbf{v}^{G1}, \mathbf{v}^{G1})}$ is equal to $\mathbf{v}^{G1}$ except for having a value of 0 for the three facts in s_t^+. As a result, this vector's norm is 1.73, which leads to a heuristic value $h(\mathbf{s_0^+}, \mathbf{s_t^+}, \mathbf{v}^g) = 1.87 - 1.73 = 0.14$. As the observation sequence does not add facts assigned a value larger than 0 by the fact observation probabilities for G2, both parts of the heuristic computation for G2 will result in the same vector norm. Hence, for G2, we have a heuristic value of 0 in this example. Consequently, FPV recognizes G1 as the most probable goal.

Table 1: Fact observation probabilities for the example depicted in Figure 1.

f	P_f^{G1}		P_f^{G2}	
	observed	not observed	observed	not observed
(is-at c1)	1.0	0.0	0.0	1.0
(is-at c2)	0.5	0.5	0.0	1.0
(is-at c3)	0.5	0.5	0.5	0.5
(is-at c4)	0.0	1.0	0.5	0.5
(is-at c5)	0.0	1.0	1.0	0.0
(is-at c6)	0.5	0.5	0.0	1.0
(is-at c7)	0.0	1.0	0.0	1.0
(is-at c8)	0.5	0.5	0.5	0.5
(is-at c9)	0.0	1.0	0.0	1.0
(is-at c10)	0.0	1.0	0.5	0.5
(is-at c11)	0.5	0.5	0.0	1.0
(is-at c12)	0.0	1.0	0.0	1.0
(is-at c13)	0.5	0.5	0.5	0.5
(is-at c14)	0.0	1.0	0.0	1.0
(is-at c15)	0.0	1.0	0.5	0.5
(is-at c16)	0.5	0.5	0.0	1.0
(is-at c17)	0.0	1.0	0.0	1.0
(is-at c18)	0.5	0.5	0.5	0.5
(is-at c19)	0.0	1.0	0.0	1.0
(is-at c20)	0.0	1.0	0.5	0.5
(is-at c21)	0.5	0.5	0.0	1.0
(is-at c22)	0.5	0.5	0.0	1.0
(is-at c23)	1.0	0.0	1.0	0.0
(is-at c24)	0.0	1.0	0.5	0.5
(is-at c25)	0.0	1.0	0.5	0.5

4 Estimating Fact Observation Probabilities

In practice, the fact observation probabilities are usually not given. Hence, this section proposes a method to estimate the fact observation probabilities from a given planning domain. To estimate fact observation probabilities, this method first determines sets of supporter actions for each possible goal. Hoffmann et al. [9] introduced supporter actions, and they can be sampled very efficiently from a Relaxed Planning Graph (RPG)[1] [9]. The idea behind supporter actions was to define a methodology for determining the actions in a given planning domain relevant to solving a given problem. More precisely, an action a is called a supporter for a fact f if $f \in add(a)$. The following subsection describes the exact algorithm that the current implementation of FPV uses to sample supporter actions. In total, FPV generates N different sets of supporter actions for each goal to ensure that the sets of supporter actions cover several possible paths. Based on the N sets of supporter actions for each goal, FPV calculates for each action a how probable it is that a occurs in one of the N sets. FPV interprets these probabilities as action observation probabilities, from which FPV subsequently derives the fact observation probabilities. Given these probabilities, FPV derives the observation probability for each $f \in F$ (i.e., $\mathbf{P}_F^g$) by calculating the probability of observing *any* of the actions for which $f \in add(a)$ holds.

4.1 Sampling Supporter Actions

To generate the N sets of supporter actions for a goal description g, FPV first generates N sets of supporter actions for each subgoal $g_i \in g$. Algorithm 2 describes the exact algorithm that FPV uses to generate the N sets of supporters for each subgoal g_i. The input of Algorithm 2 consists of four elements: f is the subgoal fact for which the algorithm samples the sets of supporter actions, RPG is

[1] An RPG is created by sequentially adding all applicable actions to the initial state in a relaxed way until all goal facts are relaxed reachable.

Algorithm 2 Supporter Action Sampling.

```
1: function SAMPLERELEVANTACTIONS(g_i, RPG, s_0, N)
2:     count ← {}                          ▷ Map from action to count in samples.
3:     samples ← []                        ▷ List of generated supporter sets.
4:     for all i ∈ range(0, N) do
5:         C ← g_i                         ▷ Facts to be supported.
6:         found ← ∅                       ▷ Stores supported facts.
7:         sups ← ∅                        ▷ Stores supporters.
8:         for t = RPG.levels to 0 do
9:             newC ← ∅                    ▷ New facts to be supported.
10:            while |C| > 0 do
11:                p ← C.pop()
12:                psups ← ∅               ▷ Potential supporters.
13:                for t2 = 0 to t do
14:                    for all a ∈ RPG.level(t2) do
15:                        if |add(a) ∩ p| > 0 then
16:                            psups ← psups ∪ {a}
17:                        end if
18:                    end for
19:                    if |psups| > 0 then
20:                        break
21:                    end if
22:                end for
23:                psups ← minCount(psups, count)
24:                a ← random(psups)
25:                found ← found ∪ {p}
26:                C ← C \ {p}
27:                sups ∪ {a}
28:                count[a] ← count[a] + 1
29:                for all n ∈ pre(a) do
30:                    if n ∉ s_0 ∧ n ∉ found ∧ n ∉ C then
31:                        newC ← newC ∪ {n}
32:                    end if
33:                end for
34:                for all r ∈ add(a) do
35:                    C ← C \ {r}
36:                    newC ← newC \ {r}
37:                end for
38:            end while
39:            C ← C ∪ {newC}
40:        end for
41:        samples ← samples.add(sups)
42:    end for
43:    return samples
44: end function
```

an RPG for the given planning problem, s_0 is the initial state, and N is the number of supporter action sets that the algorithm generates. In lines 2 and 3, Algorithm 2 initializes the $count$ map and $samples$ list. The $count$ map has actions as keys and maps each action key to the number of times the algorithm previously selected that action as a supporter action. $samples$ is the final return variable of the algorithm and stores the generated sets of supporter actions. The for loop that starts in line 4 repeats the following sampling procedure N times. For the sampling process, first the elements C, $found$, and $sups$ are initialized (lines 5-7). C is the set of facts for which the algorithm samples supporting actions, $found$ is a set that keeps track of the facts already supported, and $sups$ is the set of supporting actions already selected for the current sample. The search for supporting actions for a fact p occurs in lines 13-22. For this search, the algorithm starts at the first action level of the RPG, which ensures that it first detects supporters closer to s_0. Subsequently, the algorithm adds all potential supporter actions for p from the currently searched RPG action level to the $psups$ set. When $psups$ contains at least one element after searching the current RPG action level, the algorithm stops the search for additional supporter actions. In Line 23, the algorithm selects the potential supporter action that it selected the minimum number of times during the preceding search process (i.e., it has the minimal count value compared to all elements of $psups$). This selection procedure ensures that when different options exist for selecting supporting actions, the algorithm picks all options with roughly the same probability. When several actions have the minimum count value, the algorithm randomly selects one of them (Line 24). Afterward, the $found$, C, $sups$, and $count$ variables are updated accordingly (lines 25-28). In lines 29-33, the algorithm iterates over the selected action's preconditions and adds all facts that are not already supported or part of s_0 to $newC$. Following this, in lines 34-37, the algorithm removes all add effects of the selected action from C and $newC$ as the selected action already supports them. As a result, Algorithm 2 returns a list of sampled sets of relevant actions for reaching the subgoal g_i from s_0.

Algorithm 3 Generating Supporter Actions for g.

```
1: function RECOGNIZE GOALS(S, N, g)
2:     SG ← []                             ▷ List of supporter action sets.
3:     SG_i ← ∅                            ▷ Set of supporter actions.
4:     for all x ∈ range(0, N) do
5:         for all g_i ∈ g do
6:             s ← random(S[g_i])
7:             SG_i ← SG_i ∪ s
8:             S[g_i] ← S[g_i] \ {s}
9:         end for
10:        SG ← SG.add(SG_i)
11:        SG_i ← ∅
12:    end for
13:    return SG
14: end function
```

From these sets of supporter actions for each subgoal, FPV then generates N sets of supporter actions for g. FPV does this by using Algorithm 3. The input of Algorithm 3 is composed of three elements: The first element, S, is a map that maps all subgoals g_i to their respective list of supporter action sets sampled previously by Algorithm 2. The second input element, N, is the number of supporter action sets generated. The third input element, g, is a goal description for which the supporter action sets are generated. Initially, Algorithm 3 initializes the two sets SG and SG_i in lines 2 and 3. SG stores the generated sets of supporter actions for g, and SG_i is used as a temporary set to unify the sets of supporter actions from each subgoal g_i. In lines 4 to 12, the algorithm iterates N times to generate N sets of supporter actions from g. This is done by randomly picking one set of supporter actions for each subgoal g_i from S (Line 6) and unifying it with SG_i (Line 7). Removing all already-picked sets of supporter actions from S ensures that each set is only picked once (Line 8). After the for loop ends in Line 9, SG_i holds a complete supporter action set for g that the algorithm then adds to SG. As a result, Algorithm 3 returns a set of N supporter action sets for g.

5 Evaluation

We empirically evaluated the FPV method on 15 benchmark domains commonly used to evaluate goal recognition approaches. The empirical evaluation aims to achieve the following goals:

1. Evaluate the goal recognition precision of the FPV method compared to four existing goal recognition methods.
2. Investigate how efficient the FPV method is regarding computation time compared to four existing goal recognition methods.

All experiments consider the *online* goal recognition problem. The code used and supplementary material is publicly available [20].

Datasets. We evaluated the approaches on 15 commonly used benchmark domains (e.g., Pereira et al. [12]).

Goal Recognition Methods. To compare the performance of the FPV method to the performance of existing goal recognition methods, we implemented a planning-landmark-based approach introduced by Pereira et al. [12] (PLR), an approximate approach proposed by Ramírez and Geffner [14] (RG09), an LP Based approach introduced by Santos et al. [17] (LP), and an approach by Masters and Sardina [10] (MS). FPV, PLR, and MS were implemented in Java using the PPMAJAL[2] library as a basis for PDDL related functionalities. For the RG09 and LP approaches, we used the original code that is publicly available. We chose the PLR, LP, and MS approaches because they are recent methods for the two major types of methods that implement the PRAP principle, i.e., approaches that derive a probability distribution over the possible goals by a comparison among the costs of different plans that are calculated for each goal (MS) and approaches that use heuristic scores to rank all possible goals $g \in G$ (PLR and LP). In addition, we compare FPV to the RG09 approach because it is related to the FPV method through the use of concepts from relaxed planning. For the MS approach, we used the MetricFF planner [8] with $\beta = 1$. For the FPV method, we sample $N = 10$ relevant action sets for each g. As FPV involves some random selections, we report the results averaged over 20 repeated runs. We also tested larger values for N, which did not significantly increase recognition performance.

Experimental Design. For all experiments in this evaluation, we use machines with 24 cores, 2.60GHz, and at least 386GB RAM for all experiments in this evaluation. We used the mean goal recognition precision to assess the performance of the different methods. We calculate the precision similar to existing works (e.g., [2]). Furthermore, as we consider online goal recognition problems $R(t)$ in this evaluation, we calculated the mean precision for different fractions $\lambda = t/T$ of total observations used for goal recognition. Here, we used relative numbers because the lengths of the involved observation sequences substantially differ. Hence, the mean precision $Prec$ for a fraction $\lambda \in [0, 1]$ is calculated as follows:

$$Prec(\lambda, \mathcal{D}) = \frac{\sum_{R \in \mathcal{D}} \frac{[g_{*R} \in R(\lfloor T_R \lambda \rfloor)]}{|\mathbf{G}_{R(T_R \lambda)}|}}{|\mathcal{D}|} \quad (6)$$

Here, $\mathcal{D}$ is a set of online goal recognition problems R, g_{*R} denotes the correct goal of goal recognition problem R, T_R is the maximum value of t for online goal recognition problem R (i.e., length of observation sequence that is associated with R), and $[g_{*R} \in R(t)]$ equals one if $g_{*R} \in \mathbf{G}_{R(t)}$ and 0 otherwise, where $\mathbf{G}_{R(t)}$ is the set of recognized goals for $R(t)$. In other words, the precision quantifies the probability of picking the correct goal from a predicted set of goals $\widehat{G}$ by chance.

[2] https://gitlab.com/enricos83/PPMAJAL-Expressive-PDDL-Java-Library

Table 2: Comparison of evaluation results among all evaluated methods (MS, RG09, LP, PLR, and FPV) when applied to different benchmark domains. The precision is reported for different λ values. Column **S** reports the average spread (size of the set of goals that are recognized as the true goal) for each domain.

		Precision (for different λ)										S
		.1	.2	.3	.4	.5	.6	.7	.8	.9	1	
average	MS	.35	.43	.49	.57	.63	.67	.73	.77	.79	.81	1.9
	RG09	.32	.43	.5	.59	.65	.69	.75	.78	.82	.86	2.2
	LP	.23	.34	.42	.51	.59	.64	.7	.74	.81	.86	2.6
	PLR	.3	.35	.43	.51	.59	.66	.7	.76	.83	.9	1.2
	FPV(ours)	**.39**	**.5**	**.59**	**.66**	**.72**	**.77**	**.83**	**.87**	**.91**	**.94**	**1.1**
blocks-world	MS	.1	.13	.14	.24	.29	.36	.41	.47	.54	.55	7.1
	RG09	.08	.22	.23	.29	.38	.42	.53	.63	**.81**	**.9**	4.1
	LP	.06	.13	.15	.2	.33	.38	.46	.52	.76	.81	5.4
	PLR	.09	.09	.1	.1	.09	.09	.09	.07	.21	.27	2.8
	FPV(ours)	**.13**	**.31**	**.24**	**.32**	**.42**	**.47**	**.57**	**.66**	.8	**.9**	**1.5**
campus	MS	.73	.73	.73	.53	.57	.43	.57	.63	.6	.6	1.4
	RG09	.73	.8	.8	.83	.83	.83	.83	.8	.8	.8	1.2
	LP	.5	.87	.87	.97	.97	.97	.97	.97	.97	.97	1.2
	PLR	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	FPV(ours)	.77	**1**	**1**	**1**	.97	.97	**1**	**1**	**1**	**1**	**1**
depots	MS	**.21**	.16	.16	.22	.38	.49	.48	.54	.59	.59	1.8
	RG09	.16	**.29**	**.28**	**.32**	.45	.62	.59	.6	.61	.78	2.6
	LP	.09	.16	.17	.27	.34	.44	.57	.69	.75	.89	2.4
	PLR	.18	.14	.2	.2	.23	.3	.5	.64	.61	.82	1.2
	FPV(ours)	**.21**	.19	**.28**	**.32**	**.54**	**.69**	**.85**	**.91**	**1**	**1**	**1**
driverlog	MS	.29	**.37**	.5	**.63**	**.73**	**.84**	**.92**	**.91**	**.91**	**.96**	1.4
	RG09	.22	.27	.35	.46	.52	.6	.75	.77	.81	.81	2.9
	LP	.2	.26	.37	.5	.58	.68	.67	.68	.76	.8	2.3
	PLR	.25	.23	.39	.52	.61	.75	.8	.8	.82	.92	1.1
	FPV(ours)	**.34**	**.37**	**.58**	.61	.57	.64	.81	.86	.88	.88	**1**
dwr	MS	.13	.23	.2	.34	.34	.39	.43	.47	.46	.55	2.0
	RG09	.25	.33	.32	.26	.2	.22	.3	.27	.31	.41	2.7
	LP	.19	.18	.24	.29	.28	.33	.51	.51	.65	.81	2.0
	PLR	.23	.16	.27	**.45**	**.5**	**.54**	.52	**.63**	**.79**	**1**	1.1
	FPV(ours)	**.32**	**.47**	**.47**	.44	.36	.43	**.58**	.59	.72	.86	**1**
easy-ipc-grid	MS	.29	.32	.44	**.52**	.56	.63	**.71**	.76	.83	.94	2.4
	RG09	**.32**	**.34**	.45	.48	.57	.63	.69	.78	.85	**1**	3.0
	LP	.13	.24	.28	.35	.39	.47	.5	.57	.7	.78	3.3
	PLR	.2	.22	.36	.42	.42	.5	.58	.59	.72	**1**	**1**
	FPV(ours)	.28	.33	**.46**	.51	**.67**	**.66**	.7	**.79**	**.88**	.93	**1**
ferry	MS	.38	**.7**	**.8**	.83	.89	.95	.93	.96	.96	.96	1.2
	RG09	.28	.58	.73	.8	.89	.91	.95	.96	.96	.96	1.6
	LP	.19	.34	.54	.75	.85	.88	.91	.91	.93	.96	2.1
	PLR	.25	.38	.64	.73	.84	.89	.96	**1**	**1**	**1**	**1.1**
	FPV(ours)	**.41**	.64	.73	**.87**	**.98**	**.98**	**1**	**1**	**1**	**1**	1.2
intrusion-detection	MS	**.26**	**.45**	.59	**.8**	**.83**	**.83**	**1**	**1**	**1**	**1**	1.8
	RG09	**.26**	**.45**	.59	**.8**	**.83**	**.83**	**1**	**1**	**1**	**1**	1.8
	LP	.18	.24	.35	.38	.66	.67	.72	.78	.94	.96	2.9
	PLR	.16	.33	.34	.62	.64	.77	.81	.92	**1**	**1**	**1.1**
	FPV(ours)	.22	.4	**.65**	.77	.81	.81	**1**	**1**	**1**	**1**	**1.1**
kitchen	MS	.7	**.77**	**.87**	**.87**	.67	.67	.8	.9	**.9**	**.9**	**1.2**
	RG09	.66	.73	.77	.83	.83	.83	.83	**.93**	.8	.8	1.4
	LP	.51	.54	.58	.73	.77	.77	.7	.76	.67	.67	1.8
	PLR	.33	.33	.24	.16	.33	.33	.33	.53	.53	.53	1.5
	FPV(ours)	**.77**	**.77**	**.87**	**.87**	**.87**	**.87**	**.87**	.87	.77	.77	**1.2**
logistics	MS	.35	.41	.47	.6	.7	.73	.76	**.89**	**.97**	**1**	1.5
	RG09	.34	.47	.58	**.69**	.72	.77	.8	.82	.93	**1**	1.9
	LP	.16	.35	.41	.48	.5	.53	.68	.68	.69	.74	3.7
	PLR	.23	.39	.42	.51	.58	.76	**.83**	**.89**	**.97**	**1**	**1.2**
	FPV(ours)	**.42**	**.5**	**.6**	**.69**	**.76**	**.81**	**.83**	**.89**	**.97**	**1**	**1.2**
miconic	MS	.41	.65	.73	**.84**	.88	.86	.89	**.96**	**1**	**1**	1.3
	RG09	.32	.49	.59	.74	.82	.87	.9	.95	**1**	**1**	1.7
	LP	.24	.43	.58	.57	.76	.8	.82	.86	.95	.93	2.0
	PLR	.34	.54	.64	.71	.77	.86	.89	**.96**	**1**	**1**	**1**
	FPV(ours)	**.61**	**.77**	**.8**	.83	**.9**	**.92**	**.93**	**.96**	**1**	**1**	1.2
rovers	MS	.46	**.6**	.69	.89	.89	**.96**	**.96**	**1**	**1**	**1**	1.2
	RG09	.43	**.6**	.74	**.91**	**.92**	.93	.93	.96	.98	.98	1.5
	LP	.31	.48	.64	.67	.7	.74	.81	.84	.91	.93	2.0
	PLR	**.48**	.52	**.8**	.82	.86	**.96**	**.96**	**1**	**1**	**1**	1.1
	FPV(ours)	.45	.56	.78	.86	.89	**.96**	**.96**	**1**	**1**	**1**	**1**
satellite	MS	.35	.31	.36	.43	.59	.66	.74	.81	.81	.81	2.4
	RG09	.35	.35	.42	.54	.64	.62	.67	.71	.74	.77	2.7
	LP	.26	.29	.38	.52	.52	.62	.72	.71	.72	.75	2.9
	PLR	.34	.24	.29	.6	**.77**	.8	.82	.91	**.95**	**.96**	1.4
	FPV(ours)	**.51**	**.46**	**.58**	**.69**	.76	**.81**	**.88**	**.94**	.93	**.96**	**1.1**
sokoban	MS	**.28**	**.3**	.16	.14	.27	.3	.34	.25	.29	.3	1.2
	RG09	.15	.22	.32	.45	**.58**	.6	.61	**.68**	.73	.8	2.7
	LP	.16	.25	**.36**	**.5**	.57	**.63**	**.68**	.64	.83	.87	1.9
	PLR	.13	.21	.25	.32	.5	.57	.57	.5	**.86**	**.96**	**1**
	FPV(ours)	.08	.26	.33	.47	.54	.52	.57	.62	.7	.79	**1**
zeno-travel	MS	.32	.36	**.57**	**.68**	**.82**	**.95**	**.96**	.98	**1**	**1**	1.2
	RG09	.28	.29	.38	.48	.59	.73	.83	.89	.93	.96	1.8
	LP	.22	.3	.42	.52	.6	.65	.85	.92	.95	**1**	2.4
	PLR	.3	.41	.45	.48	.73	.82	.89	.96	**1**	**1**	**1**
	FPV(ours)	**.41**	**.45**	.54	.61	.79	**.95**	**.96**	**1**	**1**	**1**	**1**

5.1 Experimental Results and Discussion

Goal Recognition Performance. Table 2 summarizes the experimental results for all evaluated approaches when applied to the 15

Table 3: Comparison of average computation times **in seconds** among all evaluated methods when applied to all evaluated benchmark domains. The average computation time is reported for different sizes of the observation sequence (i.e., $|\mathbf{O}|$) and set of possible goals (i.e., $|G|$).

$\|G\|$		$\|\mathbf{O}\|$				
		5	10	25	50	100
5	MS	47	87	208	409	811
	RG09	14	27	68	136	272
	LP	9	18	44	87	175
	PLR	1.4	1.4	1.4	1.4	1.4
	FPV(ours)	**.29**	**.29**	**.29**	**.29**	**.29**
10	MS	94	174	416	818	1622
	RG09	28	54	136	272	544
	LP	18	35	87	175	349
	PLR	2.8	2.8	2.8	2.9	2.9
	FPV(ours)	**.57**	**.57**	**.57**	**.57**	**.57**
20	MS	188	348	832	1636	3244
	RG09	56	108	272	544	1088
	LP	35	70	175	349	698
	PLR	5.7	5.7	5.7	5.7	5.7
	FPV(ours)	**1.14**	**1.14**	**1.14**	**1.14**	**1.14**

benchmark domains. We report the average precision for different λ values. As mentioned earlier in this section, we report the averaged precision over 20 repeated runs for FPV. We also analyzed the standard deviation of FPV's recognition precision over the 20 runs and found that it is between 0 and 0.077, with an average of 0.016 for all benchmark domains. Hence, as the standard deviation is very low, we omitted it from Table 2 for better readability. Column **S** reports the average spread (size of the set of goals recognized as the true goal) for each domain. The larger this value is, the smaller the value of the precision tends to be, as predicting a larger number of goals to be the true goal generally decreases the precision measure (cf., Equation 6). The results show that the FPV method, on average, achieves the best goal recognition precision for all values of λ. Especially when dealing with low observability (i.e., $0.3 \leq \lambda \geq 0.9$), FPV significantly outperforms all other approaches. Nevertheless, it is also important to note that although FPV outperforms all other approaches on average, no single approach outperforms all other approaches in all domains. For example, the MS approach performs superior in the driverlog dataset and the PLR approach in the dwr dataset.

Computational Efficiency. Table 3 reports the extrapolated average cumulated computation time for all evaluated approaches, averaged over all 15 benchmark domains. We report the computation time *in seconds* for different potential sizes of observation sequences (i.e., $|O_t|$) and sets of possible goals (i.e., $|G|$). The values in this table are calculated from average computation time values per observation and goal for each approach and are averaged over all experiments. The table's primary purpose is to visualize how well the different recognition approaches scale regarding computation time when the size of the considered goal recognition problem is altered. The results show that the FPV approach is the most computationally efficient among all evaluated approaches. Table 3 visualizes the advantages of FPV regarding computational efficiency compared to the other evaluated approaches very well, especially when the size of the goal recognition problems is scaled. One can see that the computation time that FPV requires does not increase with an increasing number of observations. This is because once FPV has determined fact observation probabilities for each goal, FPV only has to compute the heuristic values for each goal for each new observation. The same can be observed for the PLR approach for a similar reason as for FPV. PLR only has to compute some heuristics based on the landmarks extracted upfront. The main computation time that PLR requires is due to the landmark extraction process, which is required once per online goal recognition problem, similar to the fact observation probability estimation of FPV. Nevertheless, the results show that FPV's extraction process is much more efficient than that of PLR, which results in FPV being roughly five times faster than PLR, the second fastest evaluated approach. As expected, the MS approach is the least computationally efficient, as it requires computing an entire non-relaxed plan per goal per new observation step. Interestingly, although RG09 uses relaxed plans, which can be computed more efficiently than non-relaxed plans, it also does not scale well with increasing observations. The main reason for this is most probably that the RG09 approach also recomputes a relaxed plan for each goal for each new observation step. In addition, it requires the planner to include all observations already observed in the computed plan. This results in a much more complex planning problem.

The results also show that when $|G|$ increases, the FPV and PLR approaches scale less well than when $|O|$ increases. FPV and PLR require the most computation time to compute fact observation probabilities or landmarks. With an increasing number of potential goals, this process, of course, requires more time. Nevertheless, the FPV approach still scales much better than the PLR approach and also much better than the MS, LP, and RG09 approaches, for which the scaling is even worse than for PLR.

6 Related Work

Since Ramírez and Geffner [14] introduced the idea of Plan Recognition as Planning, many approaches have adopted this paradigm [15, 16, 5, 19, 18, 10, 12, 4]. It was recognized relatively soon that the initial PRAP approaches are computationally demanding, as they require computing entire plans, especially in the case of online goal recognition. Since then, many studies addressed this problem, with the approach by Pereira et al. [12] being a recent example. This method also belongs to a recent type of PRAP method (to which FPV also belongs), which does not derive probability distributions over the set of possible goals by analyzing cost differences but ranks the possible goals by calculating heuristic values. Additional approaches from this area include a variant that Ramírez and Geffner [14] proposed as an approximation for their main approach and the Linear Programming approach that Santos et al. [17] proposed.

7 Conclusion

In conclusion, we presented FPV, a new goal recognition method that compares fact observation probabilities with the set of actually observed facts. We empirically evaluated the FPV and showed that it achieves better goal recognition precision than four state-of-the-art goal recognition methods, especially in situations of low observability. FPV is also five times more efficient regarding computation time than the second most efficient approach, PLR, and roughly 2000 times faster than the least efficient approach, MS. FPV also scales much better regarding observation sequence length than most evaluated approaches. FPV's advantage regarding computational efficiency is mainly because FPV computes the fact observation probabilities offline, which leaves minimal computational workload during online recognition. In future work, it would be interesting to consider evaluation scenarios based on actual human behavior. Another exciting path would be exploring different approaches to estimating the fact observation probabilities.

Acknowledgements

This work was funded by the German Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection (Research Grant: 67WM22003, Project: GreenPickUp) and the German Federal Ministry for Economic Affairs and Climate Action (Research Grant: 01ME23002, Project: MEDICAR 4.0).

References

[1] L. Amado, J. P. Aires, R. F. Pereira, M. C. Magnaguagno, R. Granada, and F. Meneguzzi. Lstm-based goal recognition in latent space. *arXiv preprint arXiv:1808.05249*, 2018.

[2] L. Amado, R. F. Pereira, and F. Meneguzzi. Robust neuro-symbolic goal and plan recognition. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i10.26408. URL https://doi.org/10.1609/aaai.v37i10.26408.

[3] B. Bonet, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism for planning. In *AAAI/IAAI*, pages 714–719, 1997.

[4] L. Cohausz, N. Wilken, and H. Stuckenschmidt. Plan-similarity based heuristics for goal recognition. In *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 316–321. IEEE, 2022.

[5] Y. E-Martín, M. D. R-Moreno, and D. E. Smith. A fast goal recognition technique based on interaction estimates. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 761–768. AAAI Press, 2015. ISBN 9781577357384.

[6] C. W. Geib. Problems with intent recognition for elder care. In *Proceedings of the AAAI-02 Workshop "Automation as Caregiver*, pages 13–17, 2002.

[7] C. W. Geib and R. P. Goldman. Plan recognition in intrusion detection systems. In *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, volume 1, pages 46–55. IEEE, 2001.

[8] J. Hoffmann. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *JAIR*, 20:291–341, 2003.

[9] J. Hoffmann, J. Porteous, and L. Sebastia. Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, 22:215–278, 2004.

[10] P. Masters and S. Sardina. Cost-based goal recognition for path-planning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 750–758, 2017.

[11] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl-the planning domain definition language, 1998.

[12] R. F. Pereira, N. Oren, and F. Meneguzzi. Landmark-based approaches for goal recognition as planning. *Artificial Intelligence*, 279: 103217, 2020. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2019.103217. URL https://www.sciencedirect.com/science/article/pii/S0004370219300013.

[13] D. V. Pynadath and M. P. Wellman. Accounting for context in plan recognition, with application to traffic monitoring. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, page 472–481, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1558603859.

[14] M. Ramírez and H. Geffner. Plan recognition as planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, page 1778–1783, 2009.

[15] M. Ramírez and H. Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 1121–1126. AAAI Press, 2010.

[16] M. Ramírez and H. Geffner. Goal recognition over pomdps: inferring the intention of a pomdp agent. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, page 2009–2014. AAAI Press, 2011. ISBN 9781577355151.

[17] L. R. Santos, F. Meneguzzi, R. F. Pereira, and A. G. Pereira. An lp-based approach for goal recognition as planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11939–11946, 2021.

[18] S. Sohrabi, A. V. Riabov, and O. Udrea. Plan recognition as planning revisited. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 3258–3264. AAAI Press, 2016. ISBN 9781577357704.

[19] M. Vered, G. A. Kaminka, and S. Biham. Online goal recognition through mirroring: Humans and agents. In *Annual Conference on Advances in Cognitive Systems 2016*. Cognitive Systems Foundation, 2016.

[20] N. Wilken. Fact Probability Vector Based Goal Recognition - Supplementary Material. https://github.com/nilsWilken/Fact-Observation-Probability-Based-Goal-Recognition_ECAI24.

[21] N. Wilken and H. Stuckenschmidt. Combining symbolic and statistical knowledge for goal recognition in smart home environments. In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 26–31, 2021. doi: 10.1109/PerComWorkshops51409.2021.9431145.