Knowledge-Based Systems 278 (2023) 110895

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A unified constraint-based approach for plan and goal recognition from unreliable observations

Antonio Garrido

VRAIN, Universitat Politecnica de Valencia, Camino de Vera s/n, 46022, Valencia, Spain

ARTICLE INFO

Article history:
Received 11 July 2022
Received in revised form 6 March 2023
Accepted 5 August 2023
Available online 10 August 2023

Keywords:
Plan recognition
Goal recognition
Partial observability
Noisy observations
Planning
Constraint satisfaction

ABSTRACT

Given a sequence of observations over a plan execution, plan and goal recognition are considered as interchangeable tasks in AI planning. However, strictly speaking, the former tries to identify a plan, and the latter a set of goals, that explain the observations. Both recognition tasks are data-driven, where data comprises the plan observations, and are specially useful in proactive systems. Depending on the source of knowledge about the agents under observation, these tasks are traditionally solved by two different approaches, which require a large plan library or a planning model. In between these approaches, we propose a unified novel constraint-based approach, which distinguishes between the two tasks but is valid for both. We present a formulation, based on Partial Order Causal Link planning, that is compiled from a small plan library, to approximate a model that learns the essential causality of the original planning model. We deal with unreliable observations, which include missing and noisy observations on the real world. Modeling the observations in our formulation is straightforward. The use of the learned model allows us to address a data-driven optimization task to find the plans that most satisfy those observations (plan recognition) and the goals that are sufficiently supported by the causal relationships of the observations (goal recognition). We perform a complete evaluation of our approach in IPC domains under several indicators (accuracy, spread and ROC curves) with varying degrees of partial observability and noise on the observations. We also perform a comparison with other model-based approaches from literature.

1. Introduction

Plan recognition is the task of inferring the plan, or some of its actions, of one or more agents from partial observations of their behavior [1–6], where observations are defined in terms of which actions are executed and/or which particular changes are provoked. Goal recognition is the task of recognizing agents' main goals from the observations of their interactions in an environment [7–9] and is generally considered to follow plan recognition [10].

In traditional AI planning literature, plan recognition and goal recognition are commonly interchangeable or mixed terms, and they are both understood as the problem of identifying a minimal set of top-level actions of a plan or top-level goals sufficient to explain a collection of observations [6,11,12]. However, in this paper we maintain the strict distinction between plan and goal recognition. In order to show the differences, practical applicability of both tasks and the challenges they introduce, we present a motivating example on the blocksworld domain, which has been widely used in the International Planning Competition (IPC, `www.icaps-conference.org/competitions`) and in plan/goal recognition [4,13,14]. This planning domain consists of a set of blocks, a table and a robot hand, where one block can be stacked on top of another or on the table. The top-level goal is to find a sequence of actions that achieves a final configuration of blocks or, more simply, a word.

Let us assume a simple scenario with six blocks {*A,E,R,S,T,Y*} which, for simplicity, are initially on the table. Let us assume we do not know the planning action model, but we count on a 5-plan library to achieve the goals that express the words {π_1 = *SAY*, π_2 = *TRAY*, π_3 = *TRAYS*, π_4 = *TYRES*, π_5 = *YEAR*}, where the first block of the word is the top of the stack and the last one is on the table. Each word represents a top-level atomic goal, but the candidate goals in the library are 18, e.g., {*(clear S),(on S A),(on A Y),(ontable Y),(clear T),(on T R)*...}. In particular, *(clear S)* means the block *S* has no other block on it, *(on S A)* means *S* is on top of *A*, *(ontable Y)* means *Y* is on the table, etc. Let us consider two observations Θ = {*(holding A),(holding R)*}. Thus, the only information is that the robot is holding *A*, and later *R*. These observations represent a partial sequence of predicates or actions, on an unknown plan $\pi_?$ (we assume $\pi_?$ = *RAY*). Note here that $\pi_?$ does not belong to the plan library. In our plan recognition approach, we postulate the agent is executing any plan $\pi_1, \pi_2 \ldots \pi_5$. This decision depends on which of these plans

E-mail address: agarridot@dsic.upv.es.

https://doi.org/10.1016/j.knosys.2023.110895

best explains (i.e., *maximizes the satisfaction of*) the observations. π_2 and π_3 are the only plans that satisfy the two observations of Θ, so plan recognition concludes the agent is executing π_2 or π_3. However, recognizing these plans does not necessarily mean that the pretended goal of $\pi_?$ is entirely *TRAY* or *TRAYS*, as happens in the approaches that *mix* plan and goal recognition, which assume there is only one true (hidden) atomic goal and do not deal with individual goals [4,5,13–15]. In our goal recognition approach, we postulate the agent is achieving a subset of the 18 original candidate goals. This decision depends on which goals are sufficiently explained (i.e., *above a given threshold*) by the observations. According to Θ, only {*(on R A)*, *(on A Y)*} are recognized. In π_2 and π_3, if the robot is holding *R*, *R* is finally stacked on top of *A*; and analogously with *A* on *Y*. This means the agent is trying to achieve a word that matches with *RAY*, where the symbol "$*$" represents a sequence of blocks of any size. This goal recognition shows more precision than just assuming the only one atomic goal given by plan recognition. In consequence, plan recognition and goal recognition are distinct. They should not be interchangeable as they solve different tasks and, hence, have different objectives.

Maintaining the distinction between plan and goal recognition introduces several challenges, which are specially remarkable when: (i) the planning domain is not totally known, (ii) $\pi_?$ is not part of the initial plan library, or (iii) Θ is unreliable. First, if the action model is unknown, it becomes necessary to identify, or at least to approximate, the underlying causality of the model. Second, the goals of $\pi_?$ might include individual goals from multiple plans of the library, which is frequent when dealing with multi-agent plans with non-atomic top-level goals. Third, it is important to decide when a plan/goal best explains the given observations, considering they may be unreliable. If Θ is given by malfunctioned sensors, important observations might be missing (they have not been observed when they should), and others might be noisy (they have been observed when they should not). For instance, let us now consider the possibility of noise in the observations $\Theta = \{$*(holding A)*,*(holding R)*,*(holding S)*$\}$. Plan recognition will return $\{\pi_1, \pi_2, \pi_3\}$ because these three plans satisfy two observations; a priori, *(holding R)* might be noisy in π_1, whereas *(holding S)* might be noisy in π_2 and π_3. Note that the three observations cannot be satisfied altogether in the current plan library. Goal recognition will now return {*(on S A)*, *(on R A)*, *(on A Y)*}. In particular, *(on S A)* is recognized from π_1; *(on R A)* is recognized from π_2 or π_3; and *(on A Y)* is recognized from π_1, π_2 or π_3. This means the agent is trying to achieve a word that matches with *SAY* or *RAY*.

If the library was extended with a new plan $\pi_6 = STRAY$, π_6 would be then the only plan recognized, as it satisfies the three observations. The goals recognized would be {*(on S T)*, *(on R A)*, *(on A Y)*}, which means a word that includes ST and RAY. Summing up, plan recognition returns the most similar plan/s to $\pi_?$ *w.r.t.* Θ, whereas goal recognition returns the most promising goals *w.r.t.* Θ from the goals of a set of plans that might be different but very similar to $\pi_?$.

From a reasoning perspective, plan and goal recognition can be considered as abduction processes [16], that is, reasoning to the best explanation: if we observe something, we conclude something else as the best hypothesis. By recognizing the plans or goals of other agents, we can reason about what the agents are doing, what they will do next, and what their final intentions are. This is crucial to support prediction and decision-making through data science and knowledge-based computation techniques. Plan and goal recognition are useful in a wide variety of applications that include monitoring, activity recognition, detection, anticipation and prevention, and that require proactive and intelligent responses. The most common applications include autonomous systems, computer games, smart homes and environments (specially for the elderly), human–robot interaction and collaboration, analysis of data, surveillance and warning, crisis management, crime detection and prevention, industrial control, teaching and intelligent tutoring systems, military operations, traffic monitoring, exploratory domains, and proactive personal assistants to monitor users' needs such as wheelchair movement, trajectory and manoeuvering [7,9,11,13,14,17].

From the incipient era of plan recognition [2], two kinds of plan recognition are distinguished: keyhole *vs.* intended plan recognition. In the former, the recognizer is simply observing the agent and ignores the underlying planning action model, i.e., the causal relationships between preconditions and effects of the executed actions. In the latter, the agent is cooperative and the actions are executed with the intent the planning model to be understood. In practice, this means two different approaches. The former uses an event log, a collection of plan traces or a library of plans, potentially executed by the observed agent, where the planning model is fully unknown. The recognition task consists in matching observations over the library. The latter uses a fully known planning model (or domain theory), shared between agent and observer, rather than a library of plans or policies. The recognition task consists in running planning algorithms to match a plan that is compatible with the observations. Both approaches have pros and cons (see Section 2.5 for more details). Clearly, in some real world domains it is not always possible to have a suitable, and likely large, plan library that represents the possible agents' behavior [15]. Some approaches require to encode the standard operational procedures as complex structured plans or Hierarchical Task Networks (HTN-style), which is complicated. Also, matching observations to large plan libraries tends to be costly [1]. However, in other domains it is impossible to know the planning model used by the agents because it is simply hidden. Moreover, the assumption of foreseeing such a model relies on the fact that the observations must be compatible with plans, preferably optimal or sub-optimal, executed by a somewhat rational agent [4], which is not always feasible. Also, running planning algorithms, often multiple times, is costly [13]. In between these two approaches, we propose a novel and unified approach, which automatically learns a planning model from a plan library, that is later used for both plan and goal recognition under the premises of working with unreliable observations. In short, it works as follows:

1. We create a constraint-based formulation, based on POCL (Partial Order Causal Link [18]) and [19,20], that requires a very small plan library (10 plans prove enough in our experiments) to learn a planning action model. This model is approximate and usually incomplete, as it only learns indispensable information, but it is consistent with the entire library. By consistent we mean that the learned model must satisfy the constraints imposed by all plans; i.e., if a precondition/effect is learned in an action, it must appear in every instance of such an action in any of the plans of the library. Our library is formed by simple flat plans, rather than complex HTN plans. We do not require top-level plans that are hierarchically decomposed to define a grammar of plans like in other approaches [1,11,21]. Also, we do not require action costs nor the definition of plan utilities like in other approaches [4,5,7,8,22].
2. We require a small amount of observations. Unlike typical approaches that require observations of actions [3,4,10,13–15,21], we indistinctly support the observation of predicates and/or actions; e.g., in the blocksworld domain, it is easier to observe just that one block is on top of another, rather than *stack* or *put-down* actions that involve similar movements and several predicates. We also support unreliable observations, which include both missing/partial and noisy/untrue observations.

```
(:types block)

(:predicates (on ?x - block ?y - block) (ontable ?x - block) (clear ?x - block)
             (handempty) (holding ?x - block))

(:action pick-up
 :parameters (?x - block)
 :precondition (and (clear ?x) (ontable ?x) (handempty))
 :effect (and (not (ontable ?x)) (not (clear ?x)) (not (handempty)) (holding ?x)))

(:action stack
 :parameters (?x - block ?y - block)
 :precondition (and (holding ?x) (clear ?y))
 :effect (and (not (holding ?x)) (not (clear ?y)) (clear ?x) (handempty) (on ?x ?y)))
```

Fig. 1. Example of one type, five predicates and two operators (*pick-up* and *stack*) of the blocksworld domain. The domain contains two additional operators *put-down* and *unstack*, as the inverse of *pick-up* and *stack*, respectively.

3. We use the learned model in 1 to find the plans in the library that most satisfy the observations of 2. For example, if we have learned the effect of an action, the action appears in a plan, and we observe the effect, we can infer that such observation is satisfied by that action, provided all the causal relationships in the plan imposed by the learned model are still consistent. Note this corresponds with a plan recognition task, which involves solving a maximization problem that also deals with the noisy observations. Thanks to the constraints imposed by both the model and the observations, this task is not particularly costly.
4. We use the goals of the plans recognized in 3, i.e., those that most satisfy the observations. For each goal in each plan, we analyze its causal graph [23] and match it with the causal graphs calculated for the observations, where causal graphs are derived in a POCL fashion. If there is enough overlapping in the causal graphs, which intuitively means the goal is sufficiently supported (or explained) by the causal relationships induced by the observations, the goal is marked as recognized. Note this corresponds with a goal recognition task that, eventually: (i) allows us to recognize certain goals from the plans already recognized; and (ii) reject others that are not promising *w.r.t.* the learned causal relationships.

The rest of the paper is organized as follows. Section 2 formalizes the terminology used for planning, learning, recognition and constraint satisfaction, and discusses the related work in detail. Section 3 presents our approach, organized in four steps, and provides a complete application example for clarification. In Section 4, we provide a complete evaluation of our approach under several indicators with different degrees of partial observability and noise. We also compare our approach with other model-based approaches in the literature. Finally, Section 5 concludes the paper and proposes some indications for future work.

2. Formalization and related work

This section has a twofold objective. On the one hand, we formalize the background and notation on planning, learning planning models, plan and goal recognition, and constraint satisfaction. On the other hand, we present related work and highlight the main differences *w.r.t.* our approach.

2.1. Planning

Planning is a deliberative task to build a plan of actions to achieve a set of goals [23]. In this paper, we follow the PDDL (Planning Domain Definition Language [24]) structure, based on domain, problem and plan.

Definition 1 (*Types and Predicates*)**.** PDDL defines a hierarchy of types $\mathcal{T}$ and a set of $\mathcal{T}$-parameterized predicates $\mathcal{P}$. Each predicate provides a semantic interpretation within the planning scenario and is denoted by a name applied to a sequence of zero or more parameters in $\mathcal{T}$.

Definition 2 (*Operator*)**.** An operator has a name and a set of $\mathcal{T}$-parameters. The number and type of the parameters restrict the potential predicates to be used per operator. Each operator $o \in \mathcal{O}$ has a set of preconditions ($\mathsf{pre}(o)$) and a set of add and delete effects ($\mathsf{eff}^+(o)$ and $\mathsf{eff}^-(o)$, respectively), where $\mathsf{pre}(o), \mathsf{eff}^+(o), \mathsf{eff}^-(o) \subseteq \mathcal{P}$. For simplicity, we only consider positive preconditions here, but negative ones can be simply managed by creating dummy predicates $q = \neg p$.

Definition 3 (*Domain Theory or, Simply, Domain*)**.** The planning domain is given by the tuple $\delta = \langle \mathcal{T}, \mathcal{P}, \mathcal{O} \rangle$, where the operators $\mathcal{O}$ define the planning action model.

As an example, Fig. 1 shows one type, five predicates and two operators of the blocksworld domain. For instance, given the type *block*, the predicate *(on ?x - block ?y - block)* represents the fact that block *?x* is stacked on *?y*, while *(ontable ?x - block)* represents that block *?x* is on the table. The robot uses the operator *pick-up* to hold block *?x* and *stack* to put block *?x* on top of *?y*.

Given δ, we can define a set of $\mathcal{T}$-typed constant values, which allows us to fully ground $\mathcal{P}$ and $\mathcal{O}$ into $\mathcal{V}$ and $\mathcal{A}$, respectively. $\mathcal{V}$ and $\mathcal{A}$ have all their parameters grounded and define the Boolean predicates and actions[1], respectively, to be used in a planning problem. For example, given two blocks *S* and *T*, we can ground the predicates *(on S T)*,*(on T S)* in $\mathcal{V}$, with the real meaning *(on S T)*

[1] PDDL uses the term action in the definition of the planning domain to denote an operator. To avoid confusion in our notation, we distinguish between operators, i.e., templates with parameters as represented in Fig. 1, and actions, i.e., instantiated operators where all parameters are grounded to constant values as represented in Fig. 2.

```
1: (pick-up A)
2: (stack A Y)
3: (pick-up R)
4: (stack R A)
5: (pick-up T)
6: (stack T R)
7: (pick-up S)
8: (stack S T)
```

Fig. 2. Plan of length 8 to form the word STRAY in the blocksworld domain.

$=$ *true* and *(on T S)* $=$ *true*, to be used in actions *(stack S T), (stack T S)* in $\mathcal{A}$. Since all operators in $\mathcal{O}$ are $\mathcal{T}$-parameterized, it is easy to create a mapping between the preconditions/effects of $\mathcal{O}$ and $\mathcal{A}$. An action provides the dynamics of changes in the planning domain: the action can be applied when its preconditions hold, and its effects happen after its application.

Definition 4 (*State*)**.** A state $\mathcal{S}$ is an assignment of true/false values to predicates in $\mathcal{V}$, so the size of the state space is $2^{|\mathcal{V}|}$. $\mathcal{S}$ is a full state if $|\mathcal{S}| = |\mathcal{V}|$ and a partial state if $|\mathcal{S}| < |\mathcal{V}|$. Typically, a state is modeled as a set where only the true predicates are present, while the false predicates are omitted.

Definition 5 (*Planning Problem, Aka Planning Instance*)**.** The problem is given by $\rho = \langle \delta, \mathcal{V}, \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$, where the initial state $\mathcal{I}$ is a full state ($|\mathcal{I}| = |\mathcal{V}|$), and $\mathcal{G}$ is the goal state (typically $|\mathcal{G}| \leq |\mathcal{V}|$).

Definition 6 (*Plan Trace or, Simply, Plan*)**.** A plan for ρ is defined as $\pi(\rho) = \{\langle t_1, a_1 \rangle, \langle t_2, a_2 \rangle \dots \langle t_n, a_n \rangle\}$, where each $\langle t_i, a_i \rangle$ gives the time t_i when action $a_i \in \mathcal{A}$ happens. Note that parallel plans ($t_i = t_j$ for $a_i \neq a_j$) and multiple occurrences of an action ($t_i \neq t_j$ for $a_i = a_j$) are possible. $\pi(\rho)$ induces a chronologically-ordered sequence of full states $\langle \mathcal{S}_0 \dots \mathcal{S}_{end} \rangle$, where $\mathcal{S}_0 = \mathcal{I}$ and $\mathcal{G} \subseteq \mathcal{S}_{end}$. The plan length ($length(\pi(\rho))$) is the time for $\mathcal{S}_{end}$.

As an example, Fig. 2 shows a plan to form the word STRAY in the blocksworld domain. For this plan, all the blocks are on the table in $\mathcal{I}$, and $\mathcal{G} = \{$*(clear S),(on S T),(on T R),(on R A),(on A Y),(ontable Y)*$\}$.

Definition 7 (*Causal Relationship*)**.** Given a plan $\pi(\rho)$ and two actions $a_1, a_2 \in \pi(\rho)$, being $t_1 < t_2$, a causal relationship $\langle a_1, p, a_2 \rangle$ represents the fact that a_1 supports, as one of its positive effects, the precondition p required by a_2. This relation represents the effect-precondition dependency between a_1 and a_2 in $\pi(\rho)$.

As an example, $\langle$*(pick-up A), (holding A), (stack A Y)*$\rangle$ is a causal relationship in the plan of Fig. 2 between the two first actions. As can be seen in the domain definition of Fig. 1, *(pick-up ?x)* has a positive effect *(holding ?x)* that, in this plan, supports the precondition required by *(stack ?x ?y)*.

Definition 8 (*Causal Graph*)**.** Given a plan $\pi(\rho)$, a causal graph is a directed graph where the nodes stand for the actions and the arcs for the causal relationships, or dependencies between actions.

Causal graphs are very valuable in planning as they capture the internal structure of the plans when achieving goals [23].

2.2. Learning

Learning an action model is the task of gaining knowledge by studying from experience. Particularly, we need to acquire the preconditions and effects of the operators in the domain, by identifying common structures from a library of plans or policies.

Definition 9 (*Plan Library*)**.** We define a plan library of size n, as $\Pi_n = \{\langle \mathcal{I}_1, \mathcal{G}_1, \pi(\rho_1) \rangle \dots \langle \mathcal{I}_n, \mathcal{G}_n, \pi(\rho_n) \rangle\}$. We extend each plan $\pi(\rho_i)$ in the library with its $\mathcal{I}_i$ and $\mathcal{G}_i$, which is retrieved from ρ_i[2]. Please note that we need the initial+goal states of the plans, but no intermediate states whatsoever. Also note that the plans in the library do not need to share the initial nor the goal state.

Let us consider a domain $\delta = \langle \mathcal{T}, \mathcal{P}, \mathcal{O} \rangle$, a library Π_n, and a set of empty operators $\mathcal{O}?$. $\mathcal{O}$ and $\mathcal{O}?$ are equal, but by empty we mean that their preconditions and effects are unknown and, consequently, have to be learned. We assume the name and the parameters of operators in $\mathcal{O}?$ are known, as they can be easily extracted from the plan library. Knowing this is simple: in PDDL, actions and operators share the same name, so the operators in $\mathcal{O}?$ are named from Π_n and initialized with no preconditions/effects. Such assumption, frequent in literature of learning action models [19,25,26], may seem a bit strong but it is minimally necessary to capture relationships between actions (and operators). The parameters are needed to automatically generate a set of candidate predicates that can be potentially used in every operator. For instance, "*(pick-up ?x - block){}*" and "*(stack ?x - block ?y - block){}*" are empty operators for the operators of Fig. 1.

Definition 10 (*Learning Task*)**.** Learning a planning action model is a task defined by the tuple $\mathcal{L}_n = \langle \delta, \Pi_n, \mathcal{O}? \rangle$. A solution for the learning task $\mathcal{L}_n$, denoted as $sol_{\mathcal{L}_n}$, is an approximation of $\mathcal{O}?$ to the original operators in $\mathcal{O}$, known as the reference model. It is an approximation because: (i) not all the operators in $\mathcal{O}$ are learned in $sol_{\mathcal{L}_n}$, as this depends on Π_n; (ii) some preconditions/effects in $\mathcal{O}$ may be missing in $sol_{\mathcal{L}_n}$; and (iii) some preconditions/effects not present in $\mathcal{O}$ may appear in $sol_{\mathcal{L}_n}$. Despite this, the model of preconditions+effects in $sol_{\mathcal{L}_n}$ must satisfy all plans in Π_n (completeness) and the learned model must imply no contradictions in any plan (soundness).

Intuitively, the learning task needs to complete the operators in $\mathcal{O}?$ with some predicates $\mathcal{P}$ of δ by reasoning over the plan library. This reasoning is a little inductive and a little abductive: inductive because it makes inferences based on the plans and abductive because it forms a probable, though not necessarily unique, conclusion from those plans. More formally, $sol_{\mathcal{L}_n}$ must be consistent with the information in Π_n. This means that if actions $\mathcal{A}_i$ in every $\pi(\rho_i)$ are instantiated versions of the operators in $sol_{\mathcal{L}_n}$, all plans in Π_n are consistent, that is, a correct sequence, without contradictions, of full states $\langle \mathcal{S}_{0_i} \dots \mathcal{S}_{end_i} \rangle$ is induced, where $\mathcal{S}_{0_i} = \mathcal{I}_i$ and $\mathcal{G}_i \subseteq \mathcal{S}_{end_i}$.

Fig. 3 shows an example of $sol_{\mathcal{L}_n}$ for the *pick-up* and *stack* operators. In comparison to the reference operators shown in Fig. 1, the preconditions of *pick-up* are correctly learned, but only the two first preconditions in *stack* are in the original domain. The add effects learned are correct, but other effects, in particular the delete effects, are missing.

2.3. Plan and goal recognition

Plan and goal recognition are abductive tasks to infer a set of actions and goals, respectively, that sufficiently support a sequence of observations from an unknown plan $\pi_?$.

Definition 11 (*Observation and Sequence of Observations*)**.** Given an unknown plan $\pi_?$, an observation θ is the result of watching the value of a grounded predicate in $\mathcal{V}_?$ or an action being executed in $\mathcal{A}_?$. For example, *(holding A)* is an observation in

[2] For simplicity in our notation, we also denote the library as just a collection of plans $\Pi_n = \{\pi(\rho_1) \dots \pi(\rho_n)\}$.

```
(:action pick-up
 :parameters (?x - block)
 :precondition (and (clear ?x) (ontable ?x) (handempty))
 :effect (and (holding ?x)))

(:action stack
 :parameters (?x - block ?y - block)
 :precondition (and (holding ?x) (clear ?y) (ontable ?x) (ontable ?y) (clear ?x) (handempty))
 :effect (and (on ?x ?y)))
```

Fig. 3. Example of $sol_{\mathcal{L}_n}$ for the *pick-up* and *stack* operators of the blocksworld domain. Note the two operators are an approximation of the reference operators in Fig. 1.

the first case, and *(pick-up A)* in the second case. A sequence of observations is defined as $\Theta = \{\theta_1 \dots \theta_n\}$, where the time when each θ_i is observed ($time(\theta_i)$) is unknown. Θ represents a chronological sequence because $time(\theta_i) \leq time(\theta_{i+1})$. In other words, Θ must be monotonic and preserve the ordering of the predicates/actions in $\pi_?$, that is, θ_{i+1} cannot precede θ_i in time.

Typical learning approaches only observe actions [3–5,7,10,11,13–15,21]. In addition to these observations, we also support the observation of predicates, which is less frequent [1,6]. Observing predicates is less informative but much more realistic in the real world. After all, observing the location of one block is more evident than observing the complex movement of the robot hand, which combines several predicates. In our approach, Θ can indistinctly include predicates and/or actions. Moreover, since the sensors that capture the observations may be malfunctioning, the observations are unreliable. This means that the observation of predicates/actions may be incomplete (with gaps) and/or noisy (wrongly observed).

Definition 12 (*Plan Recognition Task*)**.** A plan recognition task is defined as $\mathcal{P}_\mathcal{R} = \langle \Pi_n, sol_{\mathcal{L}_n}, \Theta \rangle$. A solution for $\mathcal{P}_\mathcal{R}$ is defined as $sol_{\mathcal{P}_\mathcal{R}} = \{\pi_i\} \subseteq \Pi_n$, where every π_i maximizes the number of observations satisfied in Θ.

It is important to note that $\pi_? \in \Pi_n$ is not required in our approach. However, if $\pi_? \in \Pi_n$ and Θ is noiseless then $\pi_? \in sol_{\mathcal{P}_\mathcal{R}}$. In such a case, the observed plan $\pi_?$ is always a plan recognized as it maximizes the number of observations satisfied. As a general case, the optimal solution for $\mathcal{P}_\mathcal{R}$ is the 1-size set $sol_{\mathcal{P}_\mathcal{R}} = \{\pi_?\}$, i.e., only the observed plan is recognized. Unfortunately, $sol_{\mathcal{P}_\mathcal{R}}$ might include more than one plan, *aka* over-recognition, which is frequent when Θ is small or noisy and, consequently, many plans satisfy the max number of observations. Therefore, guaranteeing the optimal solution is difficult.

Definition 13 (*Goal Recognition Task*)**.** A goal recognition task is defined as $\mathcal{G}_\mathcal{R} = \langle \Pi_n, sol_{\mathcal{L}_n}, \Theta \rangle$. A solution for $\mathcal{G}_\mathcal{R}$ is defined as $sol_{\mathcal{G}_\mathcal{R}} = \{g_j\} \subseteq \cup_{\mathcal{G}_{\Pi_n}}$, where $\cup_{\mathcal{G}_{\Pi_n}}$ is the union of all the problem goals achieved by plans in Π_n.

The optimal solution for $\mathcal{G}_\mathcal{R}$ is, obviously, $sol_{\mathcal{G}_\mathcal{R}} = \mathcal{G}_?$, which are the original goals achieved by $\pi_?$. Unfortunately, $sol_{\mathcal{G}_\mathcal{R}}$ might include other goals and fail to include come goals in $\mathcal{G}_?$. Therefore, guaranteeing the optimal solution is difficult.

Quality indicators. Popular metrics

Plan and goal recognition can be understood as binary classification tasks that assign a class label (recognized *vs.* not recognized) to a set of candidate plans and goals, respectively. Therefore, their quality indicators are inspired by those used in pattern recognition and machine learning classification, and keep their notation. Let *TP, FP, TN, FN* be true positive, false positive, true negative and false negative, respectively. *TP* and *TN* represent a successful recognition. In *TP* the hidden (unknown) element is recognized when it should, and in *TN* the element is predicted as not recognized because it should not be recognized. *FP* and *FN* represent a flaw in recognition. In *FP* the element is recognized when it should not, and in *FN* the hidden element is predicted as not recognized when it should be recognized. Given a number *N* of candidates, where $N = TP{+}TN{+}FP{+}FN$, we have different metrics as quality indicators:

- Accuracy, defined as *(TP+TN)/N*, is the proportion of correct predictions over the candidate elements. Accuracy = 1 means the optimal recognition, with no *FP* and *FN*. Alternatively, the error rate is defined as 1-accuracy. Some authors define a more permissive type of accuracy, which we call here *boolean accuracy*. In boolean accuracy, the result is 1 if the set of recognized elements contains the hidden element and 0 otherwise [8,13]. Note that boolean accuracy is a good indicator for *TP*, but no for *FP*. Other authors define a similar measure to accuracy, denoted as *Q*, as the proportion of problems where the hidden element is among the most likely recognized [5,7].
- Precision (positive predictive value), defined as *TP/(TP+FP)*, as the proportion of correct predictions over the recognized ones. Precision = 1 means that there are no *FP*.
- TP rate (recall or sensitivity), defined as *TP/(TP+FN)*, as the proportion of correct predictions over the hidden ones. TP rate = 1 means that there are no *FN*. Alternatively, FN rate = 1-TP rate.
- TN rate (specificity or selectivity), defined as *TN/(TN+FP)*, as the proportion of correctly not recognized elements over the elements not to be recognized. TN rate = 1 means that there are no *FP*. Alternatively, FP rate = 1-TN rate.
- Agreement Ratio (denoted as *AR*), defined as *TP/(TP+FN+FP)*, as the proportion of correct predictions over all the elements recognized together with the hidden ones. AR = 1 means that there are no flaws in recognition.
- Spread, defined as *TP+FP*, as the number of recognized elements. The spread is useful to evaluate the size of the recognized set, but no if the elements are well or badly recognized. Despite this, it is a metric commonly used, specially in plan recognition.

Although other metrics are clearly possible, such as prevalence, positive/negative likelihood ratios or F1-score, the most popular metrics used in literature are those indicated above. Many authors also use ROC (Receiver Operating Characteristic) curves to graphically compare TP rates *vs.* FP rates, or TN rates *vs.* FN rates, which provide very valuable information [13,15].

2.4. Constraint satisfaction and optimization

A Constraint Satisfaction Problem (CSP) is a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X}$ is a set of decision variables, $\mathcal{D}$ represents the domain

for each of these variables and $\mathcal{C}$ is a set of constraints among the variables in $\mathcal{X}$ that bound their values in $\mathcal{D}$.

An evaluation of values to variables is consistent if it does not violate any of the constraints in $\mathcal{C}$. A consistent evaluation is a solution if it includes values for all variables in $\mathcal{X}$. A CSP can have many solutions. If we do not define a metric over $\mathcal{X}$, we are dealing with a pure satisfaction problem, rather than a Constraint Optimization Problem (COP); i.e., many solutions are possible and equally valid. On the contrary, defining a metric allows us to optimize the value of an $\mathcal{X}$-expression. This can be used, for example, to define soft constraints, which do not need to be obligatorily satisfied, and maximize the number of them that are finally satisfied in a CSP.

2.5. Related work

Traditionally, plan (and subsequently goal) recognition has no knowledge on the planning model the agent is using. Therefore, the recognition task relies on a plan library, which is typically composed of top-level plans that are hierarchically decomposed in different ways [1,11,21]. The simplest representation is used in [11], with hierarchical plans that only include task decomposition. Then, [21] uses hierarchical plans with AND- and OR-nodes captured as grammar rules. Finally, [1] uses a more informative hierarchy, represented as a single-root directed acyclic connected graph, which includes plan steps, task decomposition, and the expected order of execution. Unlike other approaches, it supports the observation of actions and predicates. These works use abductive, Bayesian probabilistic and decision-tree methods of machine learning to generate a probability model and a mapping of observations for matching plan steps. [1,11] do not support unreliable observations, but [21] supports missing observations without noise. The overall idea is that the recognizer matches observations, as atomic instantaneous actions, to specific plan steps within the library in order to infer answers to plan recognition queries. This idea is also applied in other works. [12] deals with ordinary deductive inference based on observations, an action taxonomy, and simplicity constraints. In [27], the interpretation of observations is considered as a parsing task wherein observations are lexical tokens and plan libraries are grammars. Different types of grammars can be used, such as context-sensitive, context-free and plan tree grammars. Similarly, [28] uses Bayesian inference and abdicative probabilistic theory to map the observations onto an interpretation, together with the likelihood of the interpretation. [16] assembles the most likely observations into a Bayesian network, which is a representation of a probability distribution over the set of possible explanations or plans. [29] faces recognition as a symbolic syntactic recognition task.

Despite the success of the above systems, most of them typically focus on single-agent plan recognition. On the contrary, [22] follows a pattern recognition approach where the input plans are partial team traces, without noise, and are associated with a utility function or cost. Hence, the multi-agent plan recognition task is solved as a MAX-SATisfiability problem that tries to find the model with the maximal utility. This work addresses a maximization problem like ours. However, we look for the plans that most satisfy the observations without needing additional utilities.

Although most of the previous cited works infer a kind of graph representation of the domain in absence of noise, in terms of hierarchies or rules [1,11,12,16,21,30], and despite the importance of learning causality to understand and identify goals [31,32], they do not learn an action model to be exploited from a planning perspective. On the contrary, our approach: (i) uses a POCL planning formulation to extract causal relationships into an approximate action model from a library of multi-agent flat (PDDL) plans, notably simpler than hierarchical plans; (ii) does not require the definition of artificial plan utilities; and (iii) supports unreliable observations.

In other approaches, plan/goal recognition is considered as the inverse problem of plan synthesis [3] or, loosely speaking, it is *planning in reverse* [4,5]: the objective in planning is to seek the actions to achieve the goals, whereas the objective in plan recognition is to seek the top-level goal that explains the observations. Hence, such recognition requires a full knowledge on the agent's planning model to be able to exploit it. Under this premise, some authors transform plan recognition into a planning task, which is solved by planning algorithms. [4,5] replace the need for the plan library by an action model together with a set of candidate goals. That is, the planning model must be known by the recognizer, and the goals to be recognized are selected from an input candidate set. These works require the definition of action costs and use optimal or approximate planning algorithms to calculate the probability distribution over the candidate goals. The idea here is to find an action sequence that is a (sub-) optimal plan, in terms of the cost, for both the candidate goal and that goal extended with the observations. In [15], this model is extended to a POMDP (Partially Observable Markov Decision Process) model, where the actions are stochastic and states are partially observable. These works support partial observations, but no noise. [3] moves a step forward in the exploitation of the action model and uses a modification of a classical planning graph, as a basis to recognize the goals incrementally, where input observations can be incomplete. [7] uses the ideas of [3] to extend the work in [4] by computing cost estimates over a plan graph to infer a probability distribution over the candidate goals. [14] performs goal recognition by using event logs, where each log consists of a sequence of timestamped actions, analogous to the plan library used in our approach. Particularly, it uses process mining techniques for discovering models that describe the observed behavior and diagnosing deviations between the discovered models and observations. Unfortunately, noise is not supported in these works. [13] uses planning techniques with an emphasis on the extraction of landmarks to develop goal recognition heuristics. It supports partial observations and, although no explicit noisy observations are managed, they can be addressed as spurious actions within it. The use of landmarks may look similar to the use of causal relationships in our approach. However, landmarks and causal graphs contain different information: landmarks represent properties (or actions) that cannot be avoided to achieve a goal, whereas causal graphs represent the causal dependencies between actions. Also, the extraction and ordering of landmarks is more complex than creating a causal graph. Finally, a causal graph is never empty, but a graph formed only by conjunctive landmarks could be empty. [8] also requires artificial action costs and defines operator-counting constraints, solved by linear programming models that minimize plan costs when the observations are included. It supports partial and noisy observations.

Unlike our approach, which establishes a clear distinction between plan and goal recognition, the works in the literature typically mix these two tasks and really face a plan recognition task. However, there are two approaches that try to differentiate both tasks [6,10]. [6] introduces the idea of recognizing plans in addition to goals when dealing with unreliable observations. It assumes observations over predicates and argues that in many applications the actual actions of an agent may not be observable. It requires the action model, where actions have costs, and the set of candidate goals to assign a posterior probability to the true goal. Although this resembles a goal recognition task, it really looks for a unique true (hidden) goal, rather than for individual goals. This essentially is like plan recognition. [10]

Table 1
Plan and goal recognition in literature (references appear in chronological order).

Reference	Input	Method	PR/GR	Unreliable obs	Obs	Quality indicators	Domains tested
Goldman et al. [11]	HTN plan lib	Abductive & Probabilistic methods	PR	None	Acts	–	1
Avrahami & Kaminka [1]	HTN plan lib	Decision trees	PR	None	Acts & preds	–	1
Jigui & Minghao [3]	Action model + candidate set	Planning graphs	PR	Partial	Acts	Accuracy	3
Geib & Goldman [21]	HTN plan lib	Bayesian model	PR	Partial	Acts	–	1
Ramirez & Geffner [4]	Action model + candidate set	Planning algorithms	PR	Partial	Acts	Accuracy, *AR*, FP & FN rates	4
Ramirez & Geffner [5]	Action model + candidate set	Planning algorithms	PR	Partial	Acts	*Q*, spread	6
Pattison & Long [10]	Action model + candidate set	Planning algorithms	GR	None	Acts	Precision, TP rate	5
Ramirez & Geffner [15]	Action model + candidate set	Planning algorithms and POMDP model	PR	Partial	Acts	Accuracy, precision, ROC: TP *vs.* FP rates	4
E-Martin et al. [7]	Action model + candidate set	Plan graph & planning algorithms	PR	Partial	Acts	*Q*, spread	4
Sohrabi et al. [6]	Action model + candidate set	Planning algorithms	GR	Partial & noise	Preds	Accuracy, spread	4
Polyvyanyy et al. [14]	Event logs	Process mining techniques	PR	Partial	Acts	Precision, TP rate	5
Pereira et al. [13]	Action model + candidate set	Landmark reasoning & planning algorithms	PR	Partial & *spurious noise*	Acts	Accuracy, spread, ROC: TP *vs.* FP rates	15
Santos et al. [8]	Action model + candidate set	Linear programming	GR	Partial & noise	Acts	Boolean accuracy, spread, *AR*	4
This work	PDDL plan lib (like event logs)	Constraint optimization & causality reasoning	PR & GR	Partial & noise	Acts & preds	Accuracy, spread, ROC: TP *vs.* FP rates, TN *vs.* FN rates	12

places plan recognition before goal recognition. It also requires the action model and the set of candidate goals, and uses planning techniques to calculate the probability distribution over subsets of goals. Moreover, it assumes full observations without noise. However, it looks for the true agent's goal that is hidden to the observer which, again, is like plan recognition. On the contrary, our approach does not require: (i) a planning action model; (ii) an artificial definition of action costs; and (iii) a candidate set of plans/goals (beyond those which can be automatically extracted from the library). Additionally, our approach: (i) addresses plan and goal recognition separately; and (ii) supports unreliable observations, in the form of predicates and/or actions that can be interleaved.

To sum up this related work compilation, Table 1 depicts a summary of the main features of the most relevant works in plan and goal recognition. For comparison, the last row presents the features of our work. The features are shown in these eight columns of the table:

1. Reference of the work.
2. Source of input knowledge about the agents under observation: a plan library or an action model plus a candidate set.
3. Method used in the work.
4. If the work solves a plan recognition (PR) or a goal recognition (GR) task.
5. If the work supports unreliable observations (partial observations, observations with noise, or none).
6. If the observations are over actions, predicates or both.
7. Quality indicators, as discussed in Section 2.3. Some works ignore quality evaluation and only focus on performance, giving the number of nodes calculated, branching factors, propagation results, etc. This situation is represented by symbol "-".
8. Number of tested domains given in the work.

As can be seen in Table 1, works in the literature show some limitations. First, most works focus on plan recognition, and those that establish a difference between plan and goal recognition do not look for individual goals. Second, partial observations are supported in most works, but noisy observations are infrequent. However, noise is common in the real world, and can be seen as a measure of uncertainty. Third, the observations are mostly on actions, and only [1] supports observations on actions and predicates. Our work, as depicted in the last row, addresses all these limitations.

3. Plan and goal recognition

Our recognition method is a 4-step sequence. First, we learn a planning model from the plan library Π_n. Second, we model the observations in Θ. Third, we recognize the plans in Π_n that most satisfy the observations. Fourth, we analyze the goals of those plans to recognize the most promising goals.

3.1. Step 1. Learning a planning model

We create a POCL constraint-based formulation as a CSP to solve a learning task $\mathcal{L}_n = \langle \delta, \Pi_n, \mathcal{O}? \rangle$. Using a POCL representation allows us to explicitly model the causal relationships and the branching schemes, and to inherit the POCL completeness and soundness formal properties.

It is important to highlight that *learning* in AI usually refers to machine learning, which typically relies on artificial neural

Table 2
Formulation of variables.

Id.	Variable $\mathcal{X}$	Domain $\mathcal{D}$
X1	$\text{pre}(p, o)$	$\{false, pre\}$
X2	$\text{eff}(p, o)$	$\{false, add, del\}$
X3	$\text{sup}(p_\pi, o_\pi)$	$\emptyset \cup \{o'_\pi\} \in \pi$
X4	$\text{start}(o_\pi)$	$[0..length(\pi)]$

networks, supervised learning, deep learning, or reinforcement learning among others. In this paper, we address the learning task by solving a CSP, which conducts a search process to find the action model to be learned. This way to cope with learning is not strange, as learning action models has been addressed in the planning community by using different approaches, such as Markov logic networks, probabilistic models, MAX-SAT problems, finite state machines, genetic algorithms, constraint satisfaction and even automated planning [20].

Our formulation is inspired by [19,20], but it shows important differences and extensions. First, it is a more compact and simplified representation as we now work with a classical (non-durative) action model. Second, it is a generalization as it now learns from multiple plans, rather than just from one single plan [19]. Third, its learned model is more approximate and less precise, but it builds up faster. Fourth, it now needs to model sequences of observations at unknown times. Fifth, the observations might be noisy or incorrect, which is unsupported in [19,20].

The formulation is automatically generated from Π_n and $\mathcal{O}$?. For each $\pi \in \Pi_n$, let o_π be an action in π that is mapped onto the operator $o \in \mathcal{O}$?. It is important to recall that actions and operators share the same name, so this mapping is trivial. Let p be defined as a predicate of o, which can be potentially used as a precondition/effect, that is grounded for any action o_π, and denoted as p_π. Since we define the formulation as a CSP, we need to encode variables and constraints for every operator o with its preconditions/effects p, and the grounded versions o_π and p_π in each plan π.

3.1.1. Variables

The variables are shown in Table 2. X1 and X2 represent the variables for each $o \in \mathcal{O}$?, whereas X3 and X4 are for the grounded actions, which must be repeated per each plan $\pi \in \Pi_n$.

X1 represents whether p is, or not, a precondition in o, identified as *pre* (true) and *false*, respectively. X2 models the type of effect, and it represents if p is learned as a positive or negative effect in o (*add* or *del*, in an exclusive way); *false* means that p is not an effect of o. X3 models the supporter for the POCL causal relationship $\langle o'_\pi, p_\pi, o_\pi \rangle$: o'_π supports p_π, which is required by o_π. If $\text{pre}(p, o) = false$, p is not a precondition of o and, consequently, p_π will not be a precondition of o_π; thus, $\text{sup}(p_\pi, o_\pi) = \emptyset$, i.e., the empty supporter. Finally, X4 is the time when action o_π happens in π.

As an example, let us assume the empty operator *(pick-up ?x - block)*, the predicate *(clear ?x)* and the grounded versions *(pick-up A)* and *(clear A)*. The variables are X1 = $\text{pre}((clear\ ?x), (pick-up\ ?x\ -\ block))$, X2 = $\text{eff}((clear\ ?x), (pick-up\ ?x\ -\ block))$, X3 = $\text{sup}((clear\ A), (pick-up\ A))$ and X4 = $\text{start}((pick-up\ A))$. A solution to the corresponding CSP will learn the preconditions + effects of $\mathcal{O}$?, thus creating $sol_{\mathcal{L}_\pi}$. In our formulation, we only need to focus on the result of variables X1 and X2 for this learning. In this example, if X1 = *pre* then *(clear ?x)* is learned as a precondition of *pick-up*. If X2 = *add* then *(clear ?x)* is an add effect; and a delete effect if X2 = *del*. If the value of X1 and X2 is *false*, *(clear ?x)* is not learned as a precondition and effect of *pick-up*, respectively. As can be seen, learning the action model once the CSP is solved is straightforward.

Table 3
Formulation of constraints.

Id.	Constraint $\mathcal{C}$
C1	**if** $(\text{eff}(p, o) = del)$ **then** $\text{pre}(p, o) = pre$
C2	**if** $(\text{pre}(p, o) = pre)$ **then** $\text{eff}(p, o) \neq add$
C3	$\forall\ o$: $(\exists\ \text{pre}(p, o) \neq false)$ **and** $(\exists\ \text{eff}(p, o) \neq false)$
C4	**iff** $(\text{pre}(p, o) = false)$ **then** $\text{sup}(p_\pi, o_\pi) = \emptyset$
C5	**if** $(\text{eff}(p, o) = false)$ **then** $\forall\ o'_\pi$ that requires p_π: $\text{sup}(p_\pi, o'_\pi) \neq o_\pi$
C6	**if** $(\text{eff}(p, o) = add)$ **and** $(\text{eff}(p, o') = del)$ **then** $\text{start}(o_\pi) \neq \text{start}(o'_\pi)$
C7	**if** $(\text{sup}(p_\pi, o_\pi) = o'_\pi)$ **then** $\text{start}(o'_\pi) < \text{start}(o_\pi)$
C8	**if** $(\text{sup}(p_\pi, o_\pi) = o'_\pi)$ **and** $(\exists o^{threat} \mid \text{eff}(p, o^{threat}) = del)$ **and** $(o_\pi \neq o_\pi^{threat})$ **then** $(\text{start}(o_\pi^{threat}) < \text{start}(o'_\pi))$ **or** $(\text{start}(o_\pi^{threat}) > \text{start}(o_\pi))$
C9	$\forall\ o'_\pi$: $(\exists\ \text{sup}(p_\pi, o_\pi) = o'_\pi)$

In addition to actions in π, we include two dummy actions and operators per plan. First, init represents $\mathcal{I}$ (start(init) = 0), and has no pre and sup variables because it has no preconditions. init has as many $\text{eff}(p_i, \text{init}) = add$ as $p_i \in \mathcal{I}$ and $\text{eff}(p_j, \text{init}) = del$ for the false predicates $\neg p_j \in \mathcal{I}$, as $\mathcal{I}$ is a full state. Second, goal represents $\mathcal{G}$ (start(goal) = $length(\pi)$). goal has as many $\text{pre}(p_i, \text{goal}) = pre$ as $p_i \in \mathcal{G}$. goal has no eff variables because it has no effects.

3.1.2. Constraints

The constraints are shown in Table 3 for all variables defined in Table 2. C1–C3 represent the constraints for each operator, and C4–C9 combine constraints for operators and actions. Note that every instantiated action needs to be consistent with its mapped operator, and vice versa. In more detail, the semantics of the constraints is:

- C1 and C2 represent the classical assumption that negative effects are required as preconditions, and preconditions do not act as positive effects, respectively. These constraints are not indispensable, but they are commonly used in learning action models [25,26].
- C3 models that every operator has no empty preconditions and effects. The underlying idea is that an operator requires at least one precondition and one effect. Otherwise, it could be applied indiscriminately.
- C4 is an *if and only if* constraint that models the necessary, or unnecessary, supporter of a precondition. If p is not a precondition in o ($\text{pre}(p, o) = false$), it will no be supported in any of its grounded actions (no precondition means no need for supporter); and vice versa (every precondition needs to be supported). Clearly, if $\text{pre}((clear\ ?x), (pick-up\ ?x\ -\ block)) = pre$, $\text{sup}((clear\ A), (pick-up\ A))$ cannot be empty.
- C5 models which are the only predicates that can be used as supporters. In particular, if p is not an effect in o ($\text{eff}(p, o) = false$), then no grounded action o_π will be a supporter of p_π for any other action o'_π. Clearly, if $\text{eff}((clear\ ?x), (pick-up\ ?x\ -\ block)) = false$, *(pick-up A)* cannot appear as a supporter of *(clear A)*.
- C6 models that if two operators have contradictory/interfering effects, their grounded actions cannot happen at the same time. This avoids contradictions, because p and $\neg p$ cannot happen simultaneously.
- C7 represents the precedence defined by a causal link. It models that the supporter o'_π of p_π must happen before the requirer o_π, which is essential to satisfy the causal relationship $\langle o'_\pi, p_\pi, o_\pi \rangle$.
- C8 encodes the POCL complete way to solve a possible threat to $\langle o'_\pi, p_\pi, o_\pi \rangle$. If there exists a threatening operator o^{threat}, its grounded action o_π^{threat} cannot break the causal relationship, and it must be promoted or demoted. In other words, a threatening action o_π^{threat} cannot start between the

supporter o'_π and the requirer o_π, and must be moved forward (before the supporter) or backward (after the requirer).
- C9 represents that every action in the plan is used as a supporter at least once. This means that if an action appears in a plan, it is used in such a plan. This does not imply the plan is optimal, but it prevents actions to appear indiscriminately if they are not used.

It is important to note that C3 and C9 are not strictly part of the PDDL semantics. Although they can be easily removed if not desired, we include them in our formulation to learn more informative operators.

3.1.3. Solving the learning task

Constraint programming, CSP, COP and SATisfiability are highly related frameworks [33]. Constraint formulations can be mapped into SAT ones and vice versa. However, non-boolean variables like X2, X3 and X4 make the SAT encodings more complex, e.g., we need specific clauses to ensure that a SAT variable is given one and only one value [33]. Despite modern SAT solvers are very efficient, we are mainly interested in a flexible formulation to represent all the constraints required by $\mathcal{L}_n$, so we opt for a constraint satisfaction based solver.

We solve the learning task by using Choco (`www.choco-solver.org`), an open-source Java library for constraint programming. We deal with a satisfaction problem, so different solutions are possible. Although defining a metric over the variables in Table 2 to deal with an optimization problem is possible, we have not found a conclusive metric that leads to the best solution, i.e., the best planning action model. We have investigated different metrics, such as maximizing or minimizing the use of causal relationships and/or side effects, but the learned model is not better because of this. In other words, a more realistic action model cannot be simulated by a simple metric, which is unable to substitute the real semantics understood by the domain expert. Moreover, defining a metric has not a direct impact in reducing the variance of the learned models, as this depends on the solver and computer performance. On the contrary, we use variable+value ordering heuristics to accelerate the solving process. This is not only solver-independent, but also easily reproducible.

The variable ordering we use is X1, X2, X3, X4. The value ordering for X1 is first the *pre* value and then *false*. This tries to learn as many preconditions as possible. The value ordering for X2 is *false*, *add* and *del*, which tries to learn the minimal number of effects. For X3 and X4 we do not require a specific value ordering. These orderings give priority to a model with more preconditions and just the indispensable effects, thus returning a model $sol_{\mathcal{L}_n}$ that tends to have more preconditions and fewer effects than the original operators in δ.

3.1.4. A brief sketch on formal properties

Our formulation presents some interesting properties as already proved in [20]: polynomial size and, inherited from POCL planning, soundness and completeness.

First, the number of variables in the formulation is polynomial in the number of predicates and actions in the plans of $\mathcal{L}_n$. Let n be the number of plans in $\mathcal{L}_n$, and $|\mathcal{A}|$ and $|\mathcal{V}|$ be the max number of actions and predicates in these plans. The max number of variables is bounded by X3 and given by $O(|\mathcal{V}| * |\mathcal{A}| * n)$, which makes the number of decision variables polynomial.

Second, by soundness of the formulation, we refer to the property that it allows us to find a sound plan, where all constraints are satisfied. A plan π is sound *w.r.t.* a planning model $sol_{\mathcal{L}_n}$ if all preconditions of all actions, including the dummy ones, are satisfied when the actions happen, and no contradictions appear. This is guaranteed by the POCL [18] branching scheme of the formulation that: (i) avoids contradictory effects, which guarantees they cannot happen simultaneously (C6); (ii) encodes all the sound alternatives that support causal relationships, which guarantees that any precondition is supported before it is required (C7); and (iii) solves the possible threats by imposing an ordering, which prevents the model from breaking a causal relationship (C8). Note that this soundness means that $sol_{\mathcal{L}_n}$ represents a model that is consistent with any model able to generate the plan library.

Third, by completeness of the formulation, we mean that any possible solution can be found by solving the POCL formulation. The formulation creates a CSP that encodes the set of constraints for all plans in $\mathcal{L}_n$ that any solution $sol_{\mathcal{L}_n}$ must satisfy, and it does not discard any possible model unless it violates any of the constraints of Table 3. By the definition of the CSP and by performing a complete exploration of all its variables, a CSP solver will not discard any evaluation of values to variables unless it is inconsistent. Therefore, if a solution exists it can be found, thus guaranteeing completeness. Note that completeness means that $sol_{\mathcal{L}_n}$ represents a model that is consistent not only with any model able to generate the plan library, but also with the particular model that generated the plan library.

3.2. Step 2. Formulating observations

Once the action model $sol_{\mathcal{L}_n}$ has been learned, we deal with the observations over an unknown plan $\pi_?$. Modeling the sequence of observations $\Theta = \{\theta_1 \dots \theta_m\}$ of $\pi_?$ is straightforward in our formulation, no matter if we observe an action or a predicate. Our formulation supports mixed observations:

- If θ_i is an action, we only need to create the X3 and X4 variables like for any other action, i.e., $\mathsf{sup}(p_\pi, \theta_i)$ for its potential preconditions p_π and $\mathsf{start}(\theta_i)$.
- If θ_i is a predicate, we simply need to create a dummy action+operator obs_{θ_i} with variables X1, X3 and X4, i.e., $\mathsf{pre}(\theta_i, \mathsf{obs}_{\theta_i})$, $\mathsf{sup}(\theta_i, \mathsf{obs}_{\theta_i})$ and $\mathsf{start}(\mathsf{obs}_{\theta_i})$, as obs_{θ_i} has only one precondition (the observed predicate θ_i) and no eff variables at all.

Since we do not know the time when observations happen, we need to formulate an extra constraint C10 to preserve their chronological ordering: $\mathsf{start}(\theta_i) \leq \mathsf{start}(\theta_i + 1)\ \forall i = 1..m-1$.

3.3. Step 3. Plan recognition

The plan recognition task $\mathcal{P}_\mathcal{R} = \langle \Pi_n, sol_{\mathcal{L}_n}, \Theta \rangle$ uses the action model $sol_{\mathcal{L}_n}$ learned in step 1 all over the plans in Π_n and finds those that most satisfy the observations in Θ.

For every observation $\theta_i \in \Theta$ and plan $\pi_j \in \Pi_n$, we create a new Boolean variable X5, namely $\mathsf{sat}(\theta_i, \pi_j)$, which represents whether θ_i is satisfied, or not, in plan π_j (we assume $true = 1$ and $false = 0$). We need to formulate a new constraint C11. Since we support mixed observations:

- If θ_i is an action, mapped onto the operator o, X5 is satisfied when the following conditional constraint holds:
  if $\mathsf{pre}(p, o) = pre$ **then** $\mathsf{sup}(p_{\pi_j}, \theta_i) \neq \emptyset$ for all the potential preconditions p (C11.1).
- If θ_i is a predicate, X5 is satisfied when the following constraint holds:
  $\mathsf{sup}(\theta_i, \mathsf{obs}_{\theta_i}) \neq \emptyset$ (C11.2).

Intuitively, thanks to constraints C11.1 and C11.2, an observation is satisfied in π_j if it is consistently supported (i.e., it has a non-empty supporter) *w.r.t.* the model $sol_{\mathcal{L}_n}$. Consequently,

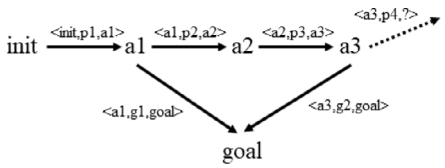


Fig. 4. Example of a causal graph used for goal recognition.

$\mathcal{P}_{\mathcal{R}}$ involves solving the general problem that maximizes the expression:

$$sol_{\mathcal{P}_{\mathcal{R}}} = \underset{\pi_j \in \Pi_n}{\text{argmax}} \sum_{\forall \theta_i \in \Theta} \text{sat}(\theta_i, \pi_j),$$

being $|sol_{\mathcal{P}_{\mathcal{R}}}| \in [1..n]$. Note that $sol_{\mathcal{P}_{\mathcal{R}}}$ requires solving n-maximization problems, one per $\pi_j \in \Pi_n$, where each of them maximizes the number of observations satisfied in π_j.

In absence of noise in Θ, the plan $\pi_?$ that produces Θ always satisfies all the observations and, hence, maximizes the previous expression. Therefore, if $\pi_? \in \Pi_n$ then $\pi_? \in sol_{\mathcal{P}_{\mathcal{R}}}$. In other words, $\pi_?$ is always a solution for $\mathcal{P}_{\mathcal{R}}$. However, other plans in the library can also maximize that expression, particularly when the number of observations is small. This means that $sol_{\mathcal{P}_{\mathcal{R}}}$ can contain multiple plans, returning a non-optimal solution. On the contrary, in presence of noise in Θ, if $\pi_? \in \Pi_n$ then $\pi_?$ is not necessarily a solution, as other plans can satisfy more (noisy) observations than $\pi_?$.

3.4. Step 4. Goal recognition

The goal recognition task $\mathcal{G}_{\mathcal{R}} = \langle \Pi_n, sol_{\mathcal{L}_n}, \Theta \rangle$ analyzes the goals of the plans in $sol_{\mathcal{P}_{\mathcal{R}}}$ of step 3 to find those that are sufficiently supported by the observations in Θ *w.r.t.* the model of causal relationships learned in $sol_{\mathcal{L}_n}$. Causal graphs identify the plan structure that satisfies the goals. Therefore, reasoning on causal graphs seems appealing to discover if a goal is sufficiently supported.

3.4.1. Motivating example. Use of causal graphs

Causal graphs can be easily generated from our formulation, as we only need to focus on the result of variables X3 of Table 2. As a motivating example, let us assume a plan $\pi \in sol_{\mathcal{P}_{\mathcal{R}}}$, with actions $\{a_1, a_2, a_3\}$ and goals $\{g_1, g_2\}$, and its causal graph depicted in Fig. 4. Building the causal graph is trivial after solving the CSP. We just need to iterate all over the X3 variables. A non-empty value for X3 (i.e., $\text{sup}(p_\pi, o_\pi) = o'_\pi$) represents the causal relationship $\langle o'_\pi, p_\pi, o_\pi \rangle$ and the arc $o'_\pi \rightarrow o_\pi$ shown in the figure. For example, we assume $\text{sup}(p_1, a_1) = \text{init}$, which represents the causal relationship $\langle \text{init}, p_1, a_1 \rangle$. We also assume $\text{sup}(g_1, \text{goal}) = a_1$ and $\text{sup}(g_2, \text{goal}) = a_3$, which represent the causal relationships $\langle a_1, g_1, \text{goal} \rangle$ and $\langle a_3, g_2, \text{goal} \rangle$, respectively. Other causal relationships are shown in the figure. For simplicity, every action in π has one precondition and only a_1 and a_3 have two effects: $\{p_2, g_1\}$ and $\{p_4, g_2\}$, respectively.

If we just observe $\Theta = \{p_4\}$, most existing approaches will fail to recognize any goal due to the lack of evidence. However, the causal graphs for the observation of p_4 ($a_1 \rightarrow a_2 \rightarrow a_3$) and for the goal g_2 ($a_1 \rightarrow a_2 \rightarrow a_3$) overlap entirely: there are 3 overlapping actions. If we focus on g_1, only a_1 overlaps. If $\Theta = \{p_3\}$ (or equivalently, a_3), its causal graph is $a_1 \rightarrow a_2$. There are 1 and 2 overlapping actions for g_1 and g_2, respectively. If $\Theta = \{p_2\}$ (or a_2), its causal graph is simply a_1, and there is 1 overlapping action for both g_1 and g_2. Intuitively, higher overlapping suggests that a goal is certainly explained by the observations. This way, we need to define a threshold to decide whether a goal is recognized or not. In this example, if the threshold is 0.5, both g_1 and g_2 will be recognized for any of the three observations, as the overlapping actions exceed the threshold. If the threshold is 1, only g_2 will be recognized when p_3 or p_4 are observed. If the threshold is 2, only g_2 will be recognized when p_4 is observed. If the threshold is 3, no goal will be recognized under these observations.

3.4.2. Calculating the threshold

Deciding the perfect threshold is not simple: high values are too restrictive and might reject real goals (false negatives), whereas low values are too loose and might recognize wrong goals (false positives). In our approach, we define the threshold as $threshold = |sups\_observations|/|sups\_goals|$, being the quotient between the number of supporters for the observations Θ, stored in the set $sups\_observations$, and the goals $\mathcal{G}$, stored in the set $sups\_goals$. These sets are automatically generated from the values of X3 after solving the CSP.

Since we support mixed observations with noise, $sups\_observations$ is the set of supporters that hold:

- $\text{sup}(p_\pi, \theta_i) \neq \emptyset$, if θ_i is an action, and being p_π a potential precondition of θ_i,
- $\text{sup}(\theta_i, \text{obs}_{\theta_i}) \neq \emptyset$, if θ_i is a predicate,

for all $\theta_i \in \Theta$. On the other side, $sups\_goals$ is the set of supporters that hold $\text{sup}(p_i, \text{goal}) \neq \emptyset$ for all $p_i \in \mathcal{G}$. In the example of Fig. 4, with just one observation (either p_4, p_3 or p_2) and two goals (g_1 and g_2), $sups\_observations = 1$, $sups\_goals = 2$, and $threshold = 1/2 = 0.5$. We can easily provide more elaborate thresholds that also take into account the size of the causal graph, but ours returns a proportion of the observations per goal, in terms of the number of supporters, and has proved very easy to calculate and adequate enough in our experiments.

3.4.3. Solving the goal recognition task

Once we have the causal graphs and the value of the threshold, we need to discover if a goal is sufficiently supported by the observations. As mentioned above, this depends on the overlapping actions for such a goal and the observations. We cannot use algorithms for graph isomorphism detection because observations are unreliable, which might provoke gaps when matching the causal graph of the goals. We have implemented a recursive procedure that counts the number of overlapping actions for the goals in $\mathcal{G}$ and the observations Θ, as shown in Algorithms 1 and 2.

We use three structures. First, *pending_sups_obs* is a set with the supporters for Θ, which is initialized with the same $sups\_observations$ used for the calculus of the threshold. Second, *visited_sups_obs*[] is a vector with the number of overlapping actions per supporter, which is initially empty. Third, *overlapping_acts*[] is a vector with the result, that is, the number of overlapping actions per goal.

Algorithm 1 initializes the structures and initiates the recursive procedure per goal by invoking Algorithm 2. The idea of Algorithm 2 is to recursively traverse the causal graph of each goal while updating the visited supporters (in *visited_sups_obs*[]) and removing them from *pending_sups_obs*. More precisely, if the supporter *act* has not been visited yet (line 3), we initialize the max number of possible overlapping actions (line 4). After updating *pending_sups_obs* in lines 5–7 (which means an overlapping has been found), we proceed recursively with the supporters of all preconditions of *act* (lines 8–11). After this recursion, line 12 calculates the number of *visited_sups_obs*[*act*] with the number of overlapping actions (broadly speaking, the max number of supporters minus the pending ones). Lines 14 and 16 finish the algorithm.

The temporal complexity of both algorithms is polynomial. Algorithm 1 is linear in the number of goals in $\mathcal{G}$. Algorithm 2 is

Algorithm 1 Calculates the overlapping actions for all goals.

```
1: procedure CalculatesOverlappingActions
Require: causal graph for 𝒢 and Θ (like in Fig. 4)
Ensure: overlapping_acts[] with the overlapping actions per goal, given Θ
2:     for all g | g is a precondition of goal do                    ▷ g ∈ 𝒢
3:         pending_sups_obs ← sups_observations
4:         visited_sups_obs[] ← ∅
5:         sup ← sup(g, goal)                                       ▷ variable X3
6:         overlapping_acts[g] ← RecCount(sup, pending_sups_obs, visited_sups_obs[])
7:     end for
8: end procedure
```

Algorithm 2 Recursively counts the number of overlapping actions.

```
1: procedure RecCount(act, pending_sups_obs, visited_sups_obs[])
2:     if act ≠ init then
3:         if act ∉ visited_sups_obs[] then                         ▷ act has not been visited yet
4:             max_size ← |pending_sups_obs|
5:             if act ∈ pending_sups_obs then
6:                 remove act from pending_sups_obs                 ▷ overlapping found
7:             end if
8:             for all p | p is a precondition of act do
9:                 new_sup ← sup(p, act)                            ▷ variable X3
10:                RecCount(new_sup, pending_sups_obs, visited_sups_obs[])
11:            end for
12:            visited_sups_obs[act] ← max_size − |pending_sups_obs|
13:        end if
14:        return visited_sups_obs[act]                             ▷ act already visited here
15:    else
16:        return 0                                                 ▷ base case: init is a dummy action
17:    end if
18: end procedure
```

linear in the number of actions in the causal graph, as every action in the graph is expanded only once (the first time it is visited in line 3).

Now that we have the overlapping actions, the goal recognition task is easy to solve. Note that in our 4-step approach, we do not need to analyze all goals in Π_n, but only those in the plans in $sol_{\mathcal{P}_\mathcal{R}}$ of step 3, thus reducing the complexity of the task. For each goal, if the overlapping value for such a goal and the observations is higher than the threshold, such a goal is marked as recognized in $sol_{\mathcal{G}_\mathcal{R}}$.

3.5. A complete application example

Let us recall the motivating example used in Section 1 with $\Pi_5 = \{\pi_1 = SAY, \pi_2 = TRAY, \pi_3 = TRAYS, \pi_4 = TYRES, \pi_5 = YEAR\}$. Although in our approach the initial state does not need to be the same in all plans, for simplicity, we consider here that all blocks are initially on the table. The plans do not need to be optimal but, also for simplicity, we assume they are the shortest. This means that only *pick-up* and *stack* are used in Π_5. For instance, $\pi_1 = \{\langle 1, (pick\text{-}up\ A)\rangle, \langle 2, (stack\ A\ Y)\rangle, \langle 3, (pick\text{-}up\ S)\rangle, \langle 4, (stack\ S\ A)\rangle\}$.

Step 1 creates the learning task $\mathcal{L}_5$ and obtains $sol_{\mathcal{L}_5}$, as the planning model of Fig. 3. This model is not the only one that can be learned. As discussed in Section 3.1.3, we cannot foresee that one model is better than another, so we use the first model found. Since *put-down* and *unstack* do not appear in Π_5, they cannot be learned. Moreover, in absence of negative preconditions/goals, there is no need to learn negative effects, so they are missing in $sol_{\mathcal{L}_5}$.[3] Although this could be a limitation in some scenarios, the model learned from just 5 plans is expressive enough and captures the essence of holding one block (*pick-up*) and placing it on top of another (*stack*).

Let us now consider an unknown plan $\pi_? = RAY$ and the observations $\Theta = \{\theta_1 = (holding\ A), \theta_2 = (holding\ R)\}$, which are formulated according to step 2. The plan recognition task of step 3 solves 5 maximization problems, one per plan in Π_5, to return $sol_{\mathcal{P}_\mathcal{R}} = \{\pi_2, \pi_3\}$, as both plans satisfy the two observations in Θ, while the other plans in Π_5 satisfy just one observation. The conclusion here, provided the two observations, is that the agent is executing π_2 or π_3. In this example, if the plan library would contain $\pi_?$, it would also satisfy the two observations and would be present in $sol_{\mathcal{P}_\mathcal{R}}$.

The number of candidate goals in $\cup_{\mathcal{G}_{\Pi_5}}$ is 18: {*(clear S)*,*(on S A)*,*(on A Y)*,*(ontable Y)*,*(clear T)*,*(on T R)*,*(on R A)*,*(on Y S)*,*(ontable S)*,*(on T Y)*,*(on Y R)*,*(on R E)*,*(on E S)*,*(clear Y)*,*(on Y E)*,*(on E A)*,*(on A R)*,*(ontable R)*}. However, we will analyze 5 goals in π_2 and 6 in π_3, which means only 7 distinct goals: {*(clear T)*,*(on T R)*,*(on R A)*,*(on A Y)*,*(ontable Y)*,*(on Y S)*,*(ontable S)*}.

The causal graph for π_2 that is created in step 4 is shown in Fig. 5. The number of supported observations and goals is 2 and 5, respectively. Thus, $threshold = 2/5 = 0.4$. The number of overlapping actions is 0 for each goal in {*(on T R)*,*(clear T)*,*(ontable Y)*} and 1 for both {*(on A Y)*,*(on R A)*}. Consequently, the only goals that are marked as recognized are {*(on A Y)*,*(on R A)*}. The causal graph for π_3 is created analogously. The number of supported observations and goals is now 2 and 6, respectively, and $threshold = 0.33$. The six goals are analyzed, but no new goals exceed the threshold; *(on A Y)* and *(on R A)* are recognized again. Note that two real goals of $\pi_?$ are not recognized: *(clear R)* and *(ontable Y)*. *(clear R)* cannot be recognized, as it does not appear in Π_5: *R* always has another block on top of it. *(ontable Y)* is a false negative that is not recognized when it should.

[3] Note that negative effects are learned when negative preconditions/goals are necessary in the actions of the library.

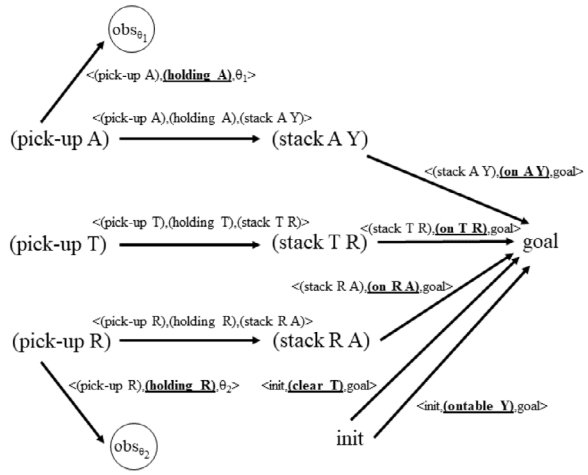


Fig. 5. Causal graph for $\pi_2 = TRAY$. obs_{θ_1} and obs_{θ_2} are the dummy actions for the observed predicates. Only the relevant causal relationships with init are represented. The goals and observations are shown as underlined text.

4. Experimental evaluation

This section provides a thorough evaluation of our approach. First, we evaluate our plan and goal recognition tasks in some IPC domains. Second, we evaluate our plan recognition task *vs.* other model-based approaches, whose results are taken from [13].

4.1. Plan and goal recognition evaluation

4.1.1. Setup

We use 12 well-known IPC domains, such as blocksworld, driverlog, elevator, floortile, grid, hanoi, etc. We have selected these domains because they can be parsed in our approach; other domains do not use types or use constants, which are unsupported functionalities in our parser. We have randomly chosen 50 problems per domain and solved them by LPG [34]. The problems do not necessarily have the same objects (e.g., in blocksworld the blocks can be different in every problem) and have different initial/goal states. Thus, the problems are unrelated and independent. The plans (up to 30 actions) are parallel, involve multiple agents and are not required to be optimal.

We use an Intel i5-6400 @ 2.70 GHz with 8 GB of RAM. The solving time in Choco is limited to 300 s for the satisfaction problem of the learning task (step 1), and to 60 s for formulating the observations and solving each maximization problem of the plan recognition task (steps 2 + 3). If no solution is found in that limit, that task is considered unsolvable. The goal recognition task (step 4) does not require the use of Choco and its solving time never exceeds 15 s.

The unreliable observations are also random. Since observing actions requires a deeper knowledge on the planning domain to understand what the agent is really doing, we always observe predicates, which is less informative but much more realistic. After all, observing an action would imply the indirect observation of its preconditions+effects. More precisely, we observe individual predicates within the sequence of states induced by the observed plan. We analyze six observability degrees, 0.05, 0.15, 0.25, 0.50, 0.75 and 1. A value of 0.05 means that the observations contain 5% of the positive predicates of the 5% of the states, i.e., we just observe the 0.25% of the predicates throughout the plan (observations on the remaining predicates are missing). A value of 1 means that all positive predicates in all the states are observed. We also consider noise in the observations, as a level of uncertainty due to a faulty read of the sensor. In particular, we analyze three values for noise, 0.0, 0.1 and 0.2. A value of 0.0 means that all the observations are flawless, but 0.1 means that 10% of the observations are faulty.

Table 4
Information on the domains, library plans and formulation size. In every domain, the first line reports the results for Π_{20} and the second line for Π_{10}.

Domain	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{X}\rvert$	$\lvert\mathcal{C}\rvert$
blocksworld	4	27	126 77	355 180	48 26	940 594	182 969 146 554
driverlog	6	28	262 124	912 436	101 59	1755 900	653 805 314 319
elevator	4	16	590 223	4817 1889	151 59	2897 1132	210 975 59 475
floortile	7	44	68 31	862 440	34 18	691 420	12 107 5181
grid	5	31	158 76	722 349	28 13	1539 818	269 778 116 228
hanoi	1	8	101 28	850 310	40 17	704 229	59 660 7173
logistics	6	24	312 148	470 222	57 28	1323 646	94 256 50 592
openstacks	3	19	159 57	767 359	75 31	1210 457	232 983 34 559
pathways	5	24	157 54	1771 688	27 13	955 393	38 015 8362
sokoban	3	33	295 191	5538 2658	44 23	3105 2066	203 996 184 220
visitall	1	5	424 240	2242 1151	279 142	2509 1431	106 924 64 984
zenotravel	5	28	101 50	409 186	53 22	588 335	22 721 16 125

We run a two-fold cross-validation evaluation, where plans are distributed into two sets: the plan library used for learning and the observed plans used for testing. We use two sizes for plan libraries, Π_{20} and Π_{10} of size 20 and 10, respectively. The size of the tests is given in each experiment. To have more meaningful results, we repeat each experiment 5 times. Every experiment uses a different plan library and observations to learn 5 different planning models and to extract average results. In order to have a clear picture on the complexity of the domains and the plans in the library, and the size of the formulation (which depends on the library), Table 4 shows the number of operators ($|\mathcal{O}|$) and predicates ($|\mathcal{P}|$, as preconditions+effects) used in each domain, and the average values for the actions ($|\mathcal{A}|$), the predicates in the initial and goal states ($|\mathcal{I}|$ and $|\mathcal{G}|$), and the variables and constraints in the formulation ($|\mathcal{X}|$ and $|\mathcal{C}|$). We manage a wide range of scenarios to increase the reliability of the results: from simple domains with just 1 operator (hanoi and visitall) and a few predicates (visitall), to more complex domains with 6–7 operators (driverlog, logistics and floortile) and dozens of predicates (floortile, sokoban, grid, etc.); and from library plans with 28 actions on average (hanoi) to 240 (visitall) in Π_{10}, and from 68 actions (floortile) to 590 (elevator) in Π_{20}.

4.1.2. Plan recognition

In order to assess the quality of our approach for plan recognition we run two experiments. The former uses Π_{20} and 5 plans for testing or observation. In the latter we use a smaller library, Π_{10}, and again 5 plans for testing. In both experiments, the observed plan is present in the library. The rationale for these experiments is to assess the accuracy, as defined in Section 2.3, in plan recognition when answering the question *which is the plan that is being executed provided the observations?* We are also

Table 5
Accuracy, spread and execution time (in seconds) for plan recognition for Π_{20} with different observability degrees (0.05...1) and noise (0.0, 0.1 and 0.2). The average and standard deviation (σ) results are shown in the last two blocks of rows.

Domain	Obs	Noise 0.0			Noise 0.1			Noise 0.2		
		Acc	Spread	Time	Acc	Spread	Time	Acc	Spread	Time
blocksworld	0.05	0.26	15.84	4.16	0.29	15.12	3.88	0.37	13.68	3.97
	0.15	0.39	13.24	4.03	0.34	14.12	3.96	0.37	13.64	3.90
	0.25	0.42	12.56	4.02	0.51	10.80	4.02	0.41	12.76	3.94
	0.50	0.63	8.40	3.99	0.60	9.08	3.96	0.52	10.56	4.12
	0.75	0.68	7.36	3.98	0.56	9.76	4.01	0.73	6.36	3.90
	1	0.76	5.88	3.95	0.80	4.92	3.99	0.77	5.52	4.00
driverlog	0.05	0.55	10.08	14.54	0.42	12.52	13.83	0.38	13.44	13.96
	0.15	0.68	7.40	14.24	0.57	9.52	13.92	0.51	10.84	14.40
	0.25	0.60	9.08	14.24	0.71	6.72	13.88	0.55	10.04	14.01
	0.50	0.82	4.68	14.14	0.85	3.92	13.84	0.74	6.16	13.84
	0.75	0.82	4.52	14.20	0.88	3.40	14.00	0.84	4.20	13.83
	1	0.87	3.60	14.14	0.91	2.80	14.13	0.88	3.36	14.16
elevator	0.05	0.39	13.28	4.78	0.45	12.00	4.60	0.35	14.08	4.57
	0.15	0.61	8.84	4.79	0.58	9.36	4.93	0.71	6.88	4.74
	0.25	0.65	7.96	4.73	0.70	6.92	4.80	0.72	6.52	4.78
	0.50	0.78	5.48	4.77	0.82	4.60	4.60	0.64	8.20	5.10
	0.75	0.84	4.24	4.81	0.85	4.08	4.62	0.93	2.40	4.62
	1	0.75	5.96	4.75	0.93	2.36	4.60	0.84	4.28	4.73
floortile	0.05	0.66	7.80	0.28	0.73	6.32	0.30	0.59	9.12	0.27
	0.15	0.70	7.08	0.28	0.68	7.32	0.27	0.71	6.76	0.27
	0.25	0.72	6.52	0.29	0.67	7.52	0.25	0.76	5.84	0.26
	0.50	0.83	4.48	0.30	0.81	4.80	0.29	0.73	6.40	0.26
	0.75	0.89	3.28	0.29	0.83	4.36	0.29	0.83	4.44	0.30
	1	0.92	2.68	0.28	0.89	3.16	0.29	0.87	3.56	0.28
grid	0.05	0.60	8.92	6.09	0.61	8.72	6.11	0.56	9.88	6.19
	0.15	0.50	10.96	6.09	0.63	8.40	6.02	0.67	7.68	5.99
	0.25	0.67	7.60	6.23	0.53	10.48	6.06	0.65	7.92	6.07
	0.50	0.87	3.60	6.30	0.85	4.08	5.95	0.85	3.92	6.07
	0.75	0.91	2.84	6.21	0.95	1.92	5.98	0.94	2.24	6.23
	1	0.97	1.60	6.36	0.97	1.52	5.97	0.95	2.08	6.18
hanoi	0.05	0.47	11.56	1.25	0.47	11.52	1.27	0.50	11.00	1.25
	0.15	0.55	9.96	1.33	0.51	10.76	1.19	0.51	10.84	1.26
	0.25	0.59	9.12	1.25	0.57	9.64	1.23	0.51	10.80	1.23
	0.50	0.69	7.12	1.27	0.61	8.72	1.24	0.68	7.32	1.25
	0.75	0.70	6.92	1.25	0.74	6.28	1.29	0.72	6.68	1.25
	1	0.85	3.92	1.25	0.80	5.00	1.22	0.78	5.36	1.22
logistics	0.05	0.56	9.80	2.22	0.60	9.00	2.27	0.55	10.08	2.33
	0.15	0.68	7.32	2.29	0.77	5.52	2.30	0.71	6.84	2.23
	0.25	0.79	5.28	2.30	0.75	6.00	2.27	0.73	6.40	2.37
	0.50	0.85	4.04	2.18	0.87	3.52	2.26	0.84	4.28	2.40
	0.75	0.93	2.44	2.20	0.87	3.68	2.18	0.91	2.80	2.20
	1	0.89	3.20	2.24	0.92	2.64	2.18	0.93	2.44	2.23
openstacks	0.05	0.61	8.80	4.90	0.52	10.64	4.65	0.56	9.76	4.76
	0.15	0.72	6.52	4.93	0.70	7.08	4.92	0.69	7.24	4.74
	0.25	0.74	6.28	5.00	0.76	5.72	4.76	0.74	6.16	4.90
	0.50	0.89	3.20	5.09	0.89	3.16	4.92	0.81	4.72	4.84
	0.75	0.93	2.40	4.98	0.91	2.88	4.69	0.89	3.16	5.01
	1	0.95	2.08	5.20	0.93	2.40	4.81	0.94	2.16	4.82
pathways	0.05	0.92	2.52	0.87	0.88	3.48	0.88	0.76	5.84	0.86
	0.15	0.76	5.80	0.88	0.73	6.48	0.86	0.68	7.36	0.87
	0.25	0.82	4.60	0.87	0.74	6.12	0.84	0.78	5.48	0.87
	0.50	0.86	3.76	0.87	0.87	3.68	0.88	0.87	3.60	0.86
	0.75	0.92	2.68	0.86	0.93	2.44	0.83	0.87	3.52	0.87
	1	0.95	1.92	0.86	0.96	1.72	0.83	0.96	1.84	0.86

(*continued on next page*)

interested in the spread, as the number of plans recognized in $sol_{\mathcal{P}_\mathcal{R}}$. The optimal value for both accuracy and spread is 1.

For each experiment, we learn a planning model from Π_{20}/Π_{10}, as defined in step 1. For each of the 5 observed plans in the testing set, we model its observations (step 2). Then, we solve the plan recognition task of step 3, which means solving a maximization problem for every plan in the library. Hence, the number of maximization problems that are solved is $5 * 20 = 100$ in Π_{20} (and $5 * 10 = 50$ in Π_{10}). We repeat each experiment 5 times, with different data, to obtain the average and the standard deviation (σ). Obviously, we are interested in low values of deviation to verify the reliability of the results.

Tables 5 and 6 depict the results for Π_{20} and Π_{10}, respectively. Focusing on Table 5, the accuracy is particularly good (between 0.50–0.75) in almost all domains for observability degrees between 0.15–0.25. It is even over 0.75 in some domains, such as logistics, pathways and sokoban. An accuracy of 0.75 means that we successfully recognize 75% of the plans, both positive and negatively. If we increase the observability degree, the accuracy is over 0.9 in some domains, such as floortile, grid, openstacks, pathways and sokoban. The spread results are as expected: when the observability degree is low (<0.50) the spread is high because there is no enough evidence to recognize the unique observed plan, and $sol_{\mathcal{P}_\mathcal{R}}$ contains many plans

Table 5 (*continued*).

Domain	Obs	Noise 0.0			Noise 0.1			Noise 0.2		
		Acc	Spread	Time	Acc	Spread	Time	Acc	Spread	Time
sokoban	0.05	0.68	7.36	4.48	0.69	7.20	4.45	0.71	6.88	4.45
	0.15	0.82	4.64	4.57	0.81	4.72	4.45	0.72	6.56	4.31
	0.25	0.89	3.24	4.39	0.87	3.56	4.32	0.87	3.60	4.54
	0.50	0.95	1.96	4.43	0.94	2.20	4.29	0.94	2.24	4.35
	0.75	0.97	1.56	4.41	0.97	1.56	4.44	0.98	1.48	4.37
	1	0.96	1.72	4.37	0.97	1.52	4.45	0.95	1.92	4.40
visitall	0.05	0.30	14.92	2.42	0.29	15.12	2.49	0.31	14.88	2.47
	0.15	0.51	10.80	2.58	0.56	9.88	2.43	0.47	11.68	2.51
	0.25	0.48	11.40	2.51	0.58	9.36	2.37	0.66	7.72	2.43
	0.50	0.80	5.08	2.41	0.83	4.32	2.32	0.76	5.72	2.32
	0.75	0.84	4.28	2.38	0.87	3.56	2.34	0.82	4.60	2.28
	1	0.76	5.76	2.39	0.87	3.64	2.33	0.95	2.08	2.34
zenotravel	0.05	0.51	10.88	0.53	0.49	11.12	0.54	0.51	10.76	0.50
	0.15	0.61	8.76	0.51	0.68	7.40	0.49	0.58	9.32	0.52
	0.25	0.75	6.04	0.54	0.66	7.76	0.50	0.71	6.76	0.51
	0.50	0.76	5.88	0.51	0.74	6.12	0.52	0.64	8.12	0.52
	0.75	0.78	5.48	0.51	0.75	5.96	0.51	0.79	5.28	0.53
	1	0.79	5.28	0.51	0.86	3.80	0.52	0.80	5.00	0.52
Average	0.05	0.54	10.15	3.88	0.54	10.23	3.77	0.51	10.78	3.80
	0.15	0.63	8.44	3.88	0.63	8.38	3.81	0.61	8.80	3.81
	0.25	0.68	7.47	3.86	0.67	7.55	3.78	0.67	7.50	3.83
	0.50	0.81	4.81	3.86	0.81	4.85	3.76	0.75	5.94	3.83
	0.75	0.85	4.00	3.84	0.84	4.16	3.77	0.85	3.93	3.78
	1	0.87	3.63	3.86	0.90	2.96	3.78	0.89	3.30	3.81
σ (std dev)	0.05	0.18	3.60	3.88	0.18	3.47	3.69	0.14	2.83	3.73
	0.15	0.12	2.46	3.80	0.13	2.56	3.73	0.12	2.39	3.83
	0.25	0.14	2.73	3.81	0.11	2.17	3.72	0.13	2.55	3.75
	0.50	0.09	1.76	3.80	0.11	2.12	3.70	0.12	2.35	3.71
	0.75	0.09	1.83	3.81	0.11	2.27	3.74	0.08	1.63	3.72
	1	0.08	1.71	3.81	0.06	1.19	3.78	0.07	1.42	3.79

Table 6
Accuracy, spread and execution time (in seconds) for plan recognition for Π_{10} with different observability degrees (0.05...1) and noise (0.0, 0.1 and 0.2). The average and standard deviation (σ) results are shown in the last two blocks of rows.

Domain	Obs	Noise 0.0			Noise 0.1			Noise 0.2		
		Acc	Spread	Time	Acc	Spread	Time	Acc	Spread	Time
blocksworld	0.05	0.25	8.52	3.56	0.36	7.36	2.99	0.31	7.92	3.19
	0.15	0.34	7.60	3.17	0.42	6.80	3.10	0.44	6.64	3.09
	0.25	0.47	6.32	3.19	0.48	6.20	3.10	0.47	6.28	3.27
	0.50	0.64	4.56	3.20	0.55	5.48	3.25	0.60	5.00	3.29
	0.75	0.75	3.52	3.19	0.68	4.24	3.17	0.70	4.04	3.30
	1	0.78	3.20	3.18	0.80	3.04	3.09	0.89	2.12	3.28
driverlog	0.05	0.35	7.48	6.49	0.39	7.08	6.64	0.43	6.72	6.53
	0.15	0.42	6.76	6.46	0.49	6.12	6.47	0.44	6.56	6.50
	0.25	0.55	5.48	6.46	0.56	5.36	6.48	0.52	5.80	6.47
	0.50	0.66	4.44	6.50	0.61	4.88	6.46	0.62	4.76	6.56
	0.75	0.76	3.36	6.53	0.82	2.80	6.57	0.74	3.64	6.51
	1	0.75	3.52	6.49	0.68	4.20	6.56	0.68	4.24	6.53
elevator	0.05	0.45	6.48	1.38	0.38	7.24	1.36	0.47	6.32	1.36
	0.15	0.61	4.92	1.40	0.54	5.60	1.41	0.59	5.12	1.43
	0.25	0.73	3.68	1.45	0.68	4.20	1.45	0.63	4.72	1.36
	0.50	0.75	3.52	1.39	0.62	4.84	1.38	0.68	4.24	1.43
	0.75	0.66	4.36	1.46	0.66	4.40	1.39	0.69	4.12	1.39
	1	0.78	3.20	1.44	0.75	3.48	1.36	0.71	3.92	1.39
floortile	0.05	0.76	3.36	0.12	0.68	4.24	0.12	0.66	4.44	0.10
	0.15	0.77	3.32	0.11	0.74	3.56	0.09	0.73	3.72	0.10
	0.25	0.76	3.36	0.10	0.76	3.36	0.09	0.74	3.60	0.11
	0.50	0.80	2.96	0.10	0.83	2.68	0.10	0.80	3.04	0.09
	0.75	0.89	2.08	0.10	0.83	2.72	0.10	0.84	2.56	0.09
	1	0.92	1.80	0.10	0.84	2.56	0.11	0.86	2.40	0.09
grid	0.05	0.51	5.88	3.76	0.62	4.84	3.67	0.56	5.40	3.77
	0.15	0.59	5.12	3.75	0.66	4.40	3.69	0.62	4.76	3.72
	0.25	0.68	4.16	3.68	0.72	3.76	3.70	0.74	3.56	3.68
	0.50	0.84	2.56	3.81	0.90	2.04	3.65	0.90	2.04	3.73
	0.75	0.94	1.56	3.86	0.92	1.80	3.67	0.91	1.88	3.69
	1	0.96	1.36	3.78	0.98	1.24	3.69	0.97	1.32	3.69

(*continued on next page*)

Table 6 (*continued*).

Domain	Obs	Noise 0.0			Noise 0.1			Noise 0.2		
		Acc	Spread	Time	Acc	Spread	Time	Acc	Spread	Time
hanoi	0.05	0.56	5.44	0.16	0.63	4.72	0.14	0.58	5.16	0.15
	0.15	0.57	5.32	0.16	0.54	5.64	0.15	0.52	5.84	0.16
	0.25	0.63	4.68	0.18	0.56	5.44	0.16	0.52	5.76	0.14
	0.50	0.62	4.84	0.16	0.59	5.12	0.15	0.66	4.40	0.14
	0.75	0.76	3.44	0.15	0.75	3.52	0.15	0.69	4.08	0.15
	1	0.82	2.76	0.16	0.82	2.84	0.15	0.79	3.08	0.13
logistics	0.05	0.64	4.64	2.30	0.58	5.24	2.31	0.58	5.16	2.29
	0.15	0.69	4.12	2.33	0.70	4.04	2.30	0.63	4.72	2.30
	0.25	0.69	4.08	2.28	0.65	4.48	2.32	0.76	3.36	2.30
	0.50	0.80	3.04	2.32	0.88	2.24	2.30	0.86	2.36	2.26
	0.75	0.85	2.48	2.29	0.86	2.40	2.28	0.83	2.72	2.24
	1	0.90	2.00	2.32	0.87	2.28	2.26	0.91	1.92	2.27
openstacks	0.05	0.66	4.36	0.72	0.61	4.88	0.73	0.64	4.56	0.79
	0.15	0.62	4.84	0.81	0.66	4.40	0.76	0.62	4.76	0.71
	0.25	0.68	4.24	0.84	0.62	4.80	0.77	0.62	4.80	0.75
	0.50	0.79	3.12	0.84	0.77	3.28	0.73	0.70	3.96	0.72
	0.75	0.88	2.20	0.76	0.92	1.84	0.73	0.90	2.04	0.73
	1	0.91	1.88	0.78	0.92	1.76	0.72	0.92	1.80	0.75
pathways	0.05	0.84	2.56	0.19	0.76	3.36	0.20	0.78	3.24	0.20
	0.15	0.86	2.44	0.18	0.87	2.32	0.18	0.85	2.52	0.20
	0.25	0.86	2.40	0.19	0.84	2.56	0.18	0.86	2.40	0.19
	0.50	0.89	2.08	0.19	0.91	1.88	0.19	0.85	2.48	0.18
	0.75	0.97	1.32	0.17	0.90	1.96	0.18	0.90	2.04	0.19
	1	0.97	1.32	0.19	0.96	1.40	0.19	0.92	1.80	0.17
sokoban	0.05	0.67	4.28	4.09	0.65	4.48	3.89	0.66	4.40	3.98
	0.15	0.80	2.96	4.04	0.81	2.88	3.95	0.82	2.84	3.94
	0.25	0.83	2.72	4.09	0.88	2.24	3.95	0.87	2.32	3.92
	0.50	0.92	1.84	3.98	0.90	2.00	3.83	0.90	1.96	3.88
	0.75	0.92	1.76	3.99	0.94	1.64	3.88	0.93	1.72	4.00
	1	0.93	1.72	3.93	0.94	1.60	3.94	0.94	1.56	3.94
visitall	0.05	0.23	8.68	1.45	0.32	7.76	1.45	0.37	7.28	1.44
	0.15	0.46	6.44	1.43	0.49	6.08	1.44	0.58	5.20	1.51
	0.25	0.43	6.72	1.40	0.60	5.04	1.45	0.70	4.00	1.46
	0.50	0.70	3.96	1.42	0.82	2.84	1.42	0.88	2.20	1.43
	0.75	0.85	2.48	1.41	0.86	2.40	1.40	0.94	1.56	1.38
	1	0.88	2.24	1.44	0.93	1.68	1.41	0.96	1.36	1.41
zenotravel	0.05	0.45	6.52	0.38	0.46	6.44	0.41	0.30	7.96	0.38
	0.15	0.46	6.40	0.36	0.37	7.28	0.37	0.45	6.48	0.35
	0.25	0.46	6.44	0.38	0.52	5.80	0.34	0.50	6.04	0.35
	0.50	0.60	5.04	0.37	0.62	4.76	0.34	0.49	6.12	0.35
	0.75	0.62	4.80	0.35	0.74	3.64	0.37	0.70	4.00	0.33
	1	0.70	4.00	0.36	0.66	4.36	0.34	0.73	3.68	0.35
Average	0.05	0.53	5.68	2.05	0.54	5.64	1.99	0.53	5.71	2.01
	0.15	0.60	5.02	2.02	0.61	4.93	1.99	0.61	4.93	2.00
	0.25	0.65	4.52	2.02	0.66	4.44	2.00	0.66	4.39	2.00
	0.50	0.75	3.50	2.02	0.75	3.50	1.98	0.75	3.55	2.01
	0.75	0.82	2.78	2.02	0.82	2.78	1.99	0.81	2.87	2.00
	1	0.86	2.42	2.01	0.85	2.54	1.99	0.86	2.43	2.00
σ (std dev)	0.05	0.19	1.95	2.03	0.15	1.46	2.00	0.15	1.52	2.00
	0.15	0.16	1.61	1.99	0.16	1.57	1.98	0.14	1.37	1.99
	0.25	0.14	1.44	1.99	0.13	1.26	1.98	0.14	1.39	1.99
	0.50	0.11	1.07	2.00	0.14	1.40	1.98	0.14	1.38	2.01
	0.75	0.11	1.11	2.02	0.10	0.96	2.00	0.10	1.03	2.01
	1	0.09	0.89	1.99	0.11	1.07	2.00	0.10	1.03	2.01

(over-recognition). The spread gets lower, but still higher than 1, when the observability degree increases. The fact of using random observations sometimes produces odd results; e.g., an increase in the observability degree may deteriorate a little the accuracy and spread. This happens punctually in a few domains, but it does not happen on average. Regarding the standard deviation, the results are very good for the accuracy. The accuracy is always bounded by 1, so the deviation is small here. In other words, the data are very little dispersed in relation to the average. The standard deviation for the spread is good, taking into consideration that the results are not bounded. The standard deviation for the time is more variable, because the execution time highly depends on the domain: the values range from less than 1 s (e.g., floortile, pathways and zenotravel) to almost 15 s (e.g., driverlog).

The inclusion of noise in the observations has no significant impact in the results. Plan recognition involves maximization problems where some observations are not finally satisfied and, therefore, simply ignored and useless. We have detected that most of the ignored observations are the noisy ones. In consequence, a higher value of noise does not necessarily deteriorate the accuracy and spread. Actually, we have realized that when the observability degree is high, noisy observations can help better discern the observed plan and discard other plans in some domains, which slightly leads to better results. We have noted that our approach is specially useful to discard the plans that do not satisfy the induced constraints. Due to the constraint-based approach, we are better discarding plans that are not being executed,rather than returning in $sol_{\mathcal{P}_{\mathcal{R}}}$ the only plan that is being

executed. In other words, our approach tends to be somewhat conservative and returns non-optimal solutions, i.e., $sol_{\mathcal{P}_\mathcal{R}}$ has more than one plan. In any case, the difference in accuracy for different values of noise is around 0.05 on average, which can be considered irrelevant given the random unreliable observations. The tendency in the spread is similar. As for the standard deviation, the values for the accuracy and time are barely affected by the noise, whereas the values for the spread are just a little more affected.

Finally, as for hypothesis testing, we have conducted the ANalysis Of VAriance (ANOVA) to test whether there exists a statistically significant difference between the results when different values of noise are used. Three ANOVAs have been calculated (accuracy, spread and time), in which three populations have been tested (with noise 0.0, 0.1 and 0.2) per ANOVA. In all cases, there are no differences in the population means with a confidence interval of 99%, which means the value of noise is no significant.

If we focus on Table 6, the accuracy results are very similar to Table 5. This means that our approach does not need large plan libraries for good accuracy, and Π_{10} is enough to get high values of accuracy. Actually, the accuracy obtained with Π_{20} is not significantly better than with Π_{10} (0.03 on average and 0.06 at most). From a statistical perspective, we have calculated the ANOVA for the accuracy with two populations (Π_{20} and Π_{10}) and there are no differences in the population means with a confidence interval of 99%. The results for the spread and time in Π_{10} follow similar tendencies to Π_{20}. But, obviously, the values for Π_{10} are not comparable in absolute terms to those for Π_{20}: the spread and time are now smaller because there are fewer plans in the library. The nature of the random observations still produces some punctual inconsistent results, but the average tendency is very consistent: when the observability degree increases, the accuracy and spread results are monotonically better.

The standard deviation follows the same tendency shown in Table 5: it is very small for the accuracy, a bit higher for the spread and even higher for the time. The deviation in the accuracy is very similar in both Π_{20} and Π_{10}, as the accuracy remains bounded. The deviation for the spread and time are better in Π_{10} than in Π_{20}. Clearly, using now a smaller plan library with only 10 plans helps reduce the dispersion of the data.

The inclusion of noise is no significant, and the results remain very similar under noisy observations. The use of a smaller plan library means that the results are less affected by the noise in terms of the standard deviation. Again, we have calculated three ANOVAs (accuracy, spread and time) with three populations each (noise 0.0, 0.1 and 0.2). In all cases, there are no differences in the population means with a confidence interval of 99%, which verifies the value of noise is no significant.

Finally, in both tables, the optimal value of 1 for accuracy and spread is highly unlikely. From the accuracy perspective, the TN are usually incomplete: we fail to discard some plans. From the spread perspective, there are FP: we are conservative and recognize some plans that we should not. On the contrary, the TP values are very good. In all our experiments, the observed plan is always recognized, i.e., it belongs to $sol_{\mathcal{P}_\mathcal{R}}$, in absence of noise. When noise is considered, it is recognized in over 97% of the experiments. This corresponds with 1 and 0.97 values for the boolean accuracy used in some works [8,13].

ROC analysis. In Tables 5 and 6 we mainly focus on positive predictions, but in a recognition task we also need to assess the false predictions, in particular the comparison of TP *vs.* FP rates and TN *vs.* FN rates to analyze the trade-off between true–false positive results and true–false negative results. We do this by adapting the notation of a ROC curve as a representation for binary classification. More specifically, we combine the results of the 12 domains into a cloud of 24 points (12 of Π_{20} and 12 of Π_{10}) per each observability degree (0.05...1) and plot them in the ROC space, which graphically shows the performance of the plan recognition task by evaluating TP *vs.* FP rates (Fig. 6)

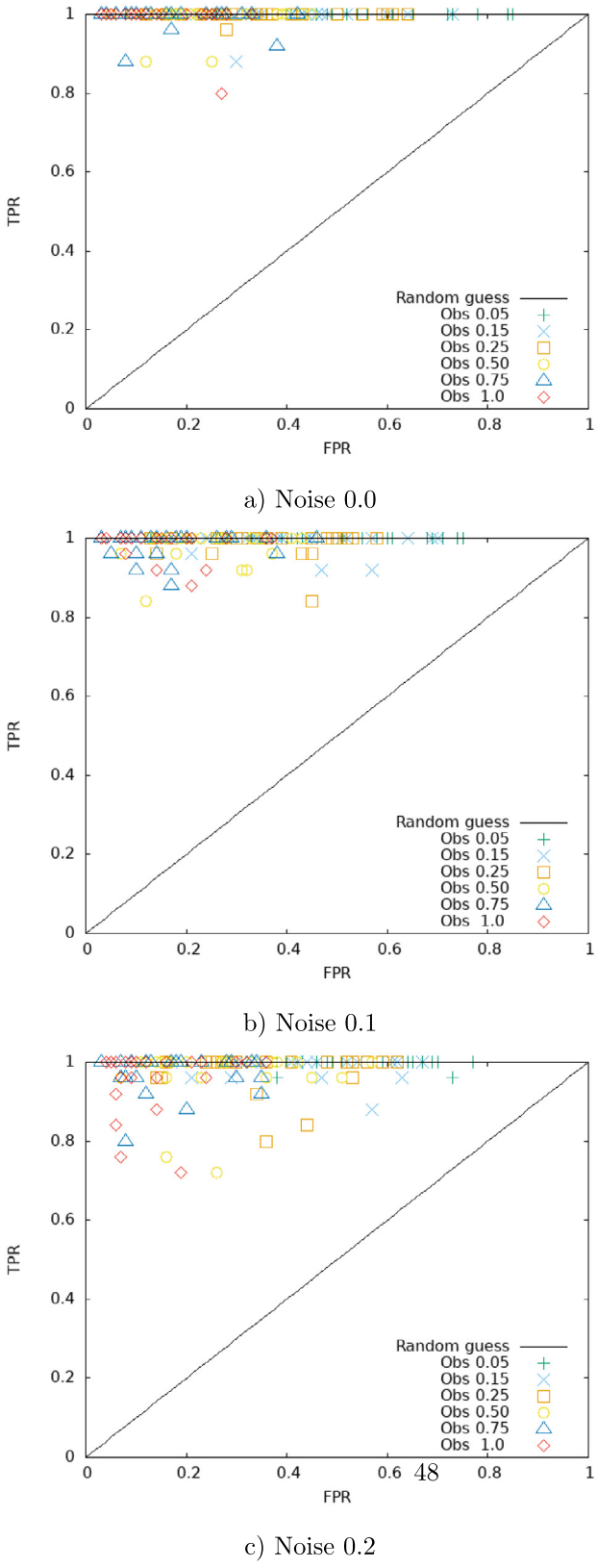


Fig. 6. Comparative ROC curve for plan recognition with different observability degrees: TP rate *vs.* FP rate with the three values of noise.

and TN *vs.* FN rates (Fig. 7). In the ROC curve, the diagonal line represents a random guess to recognize (positive or negatively) a plan from observations. The points above the diagonal represent a successful classification or better than random, whereas points below the diagonal represent bad results or worse than random. Consequently, we want to have points above the diagonal. The optimal recognition is a point in the upper left corner, that is, coordinate $X = 0$ (no false rate) and $Y = 1$ (a true rate). The closer a point to that corner is, the better the plan recognition task is.

Focusing on Fig. 6, the results are good: the TP rate (recall) is very close to 1, no matter the observability degree, whereas the FP rate remains low and only increases for observability degrees between 0.05–0.25. This means the cloud of points is closer to the $Y = 1$ line. We can note here that increasing the noise has more impact than in the accuracy/spread of Tables 5 and 6. Particularly, we can see how the points slightly tend to the diagonal when the noise value increases.

If we focus on Fig. 7, the results are also interesting: there are high values of the TN rate (specificity) and low values of the FN rate, which means the cloud of points are closer to the $X = 0$ line. Again, when the noise is increased, the points get slightly close to the diagonal.

As a summary, all the points in both figures are above the diagonal, which is a good result. Also, we can see that this approach for plan recognition is specially good to detect TP and to reduce FN, i.e., to recognize the observed plan and to reduce flaws in recognition.

4.1.3. Goal recognition

In order to assess the quality of our approach for goal recognition we also run two experiments. The former uses Π_{20} and 20 plans for testing, whereas the latter uses Π_{10} and 30 plans for testing. Since we are now only interested in the goals, the observed plan used for testing is not present in the plan library. We assess the accuracy when answering the question *which are the individual goals the agent is trying to achieve provided the observations?* The accuracy is now defined as $(TP+TN)/candidate\_goals$, where *candidate_goals* is $\cup_{\mathcal{G}_{\Pi_{20}}}$ or $\cup_{\mathcal{G}_{\Pi_{10}}}$ in every experiment, that is, the union of all distinct goals achieved in Π_{20} and Π_{10}, respectively. The size of $\cup_{\mathcal{G}_{\Pi_{20}}}$ ranges within [7-28], and within [6-26] in $\cup_{\mathcal{G}_{\Pi_{10}}}$. We are also interested in the spread. Unlike plan recognition, where the optimal spread is 1, the number of *real goals* to be recognized in the goal recognition task is variable and higher than 1. Therefore, the original definition of spread is not really useful now. Instead, we define a new indicator: *%Spread=Spread/real goals*. If *%spread* is lower than 1, some real goals are not recognized (under-recognition). If it is higher than 1, the task has recognized more goals than the real ones (over-recognition). If it is 1, the task has recognized the same number of goals than the real ones, although this does not mean the goals are always well recognized (recall that the spread is given by $TP+FP$). The optimal value for both accuracy and %spread is 1.

For each experiment, we learn a planning model from Π_{20}/Π_{10}, as defined in step 1. For each of the 20/30 observed plans in the testing set, we model its observations (step 2). The number of maximization problems that are necessary in step 3, previously to goal recognition (step 4), is $20 * 20 = 400$ in Π_{20} (and $30 * 10 = 300$ in Π_{10}). We repeat each experiment 5 times, with different data, to obtain average results.

Tables 7 and 8 depict the results for Π_{20} and Π_{10}, respectively. Focusing on Table 7, the accuracy is specially good (greater than 0.7) in almost all domains from the minimal observability degree of 0.05. It is around 0.85, and even more, in some domains, such as blocksworld, grid, hanoi, pathways and zenotravel. An accuracy of 0.85 means that we successfully recognize 85% of the goals, both positive and negatively. The %spread is, on average, close to 1 but in a few domains it is lower (floortile and pathways) or higher (grid, hanoi and visitall). We have analyzed these punctual

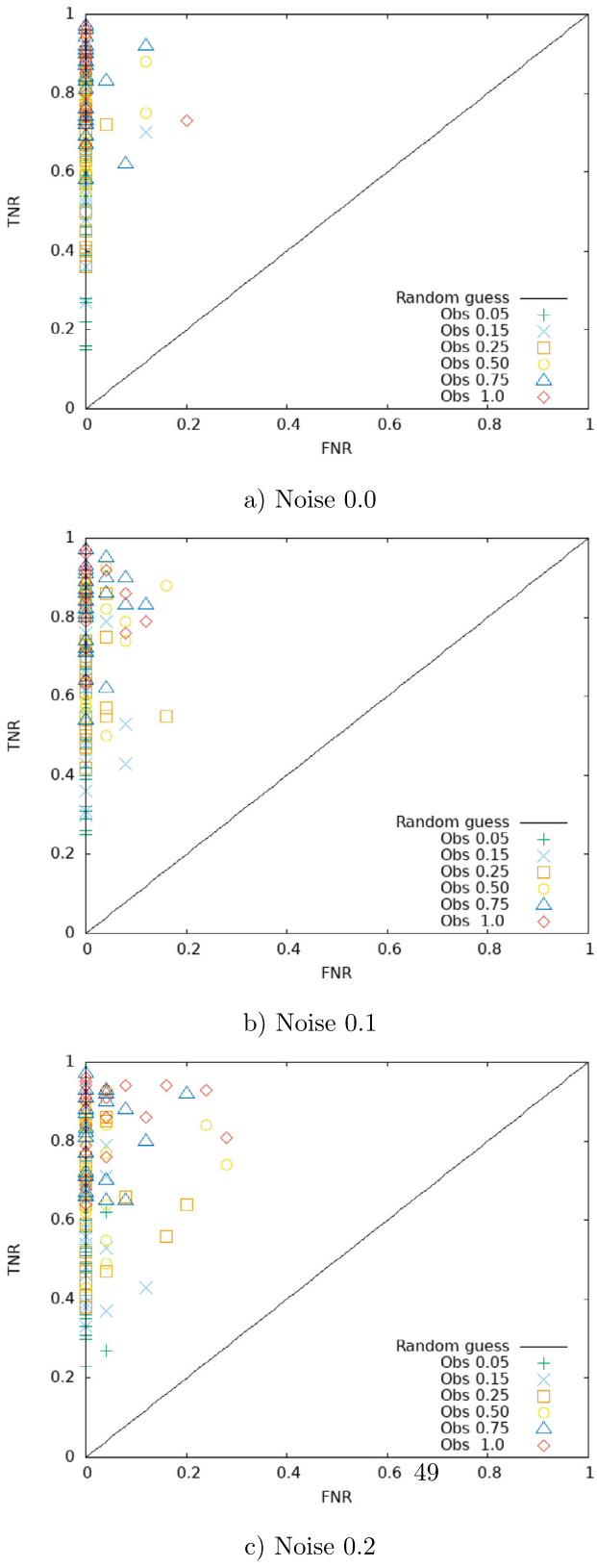


Fig. 7. Comparative ROC curve for plan recognition with different observability degrees: TN rate *vs.* FN rate with the three values of noise.

Table 7
Accuracy, %spread and execution time (in seconds) for goal recognition for $\mathit{\Pi}_{20}$ with different observability degrees (0.05...1) and noise (0.0, 0.1 and 0.2). The average and standard deviation (σ) results are shown in the last two blocks of rows.

Domain	Obs	Noise 0.0			Noise 0.1			Noise 0.2		
		Acc	%Spread	Time	Acc	%Spread	Time	Acc	%Spread	Time
blocksworld	0.05	0.85	0.48	3.68	0.84	0.54	3.59	0.84	0.63	3.62
	0.15	0.83	1.00	3.53	0.82	0.87	3.63	0.83	0.81	3.66
	0.25	0.85	0.88	3.45	0.84	0.78	3.76	0.82	0.91	3.63
	0.50	0.85	0.81	3.59	0.85	0.89	3.68	0.85	0.82	3.73
	0.75	0.85	0.90	3.68	0.84	0.74	3.82	0.85	0.80	3.63
	1	0.84	0.85	3.63	0.85	0.90	3.65	0.85	0.80	3.74
driverlog	0.05	0.73	0.25	12.44	0.71	0.32	12.81	0.72	0.30	12.87
	0.15	0.69	0.52	12.43	0.69	0.56	12.83	0.68	0.53	12.88
	0.25	0.67	0.75	12.57	0.65	0.80	12.76	0.66	0.76	12.86
	0.50	0.62	1.06	12.69	0.64	0.96	12.72	0.64	0.94	12.82
	0.75	0.64	1.06	12.50	0.65	1.05	12.87	0.65	0.96	12.88
	1	0.65	1.00	12.32	0.64	1.01	12.78	0.66	0.98	12.88
elevator	0.05	0.50	0.49	4.11	0.52	0.52	4.24	0.52	0.51	4.39
	0.15	0.79	0.80	4.40	0.79	0.80	4.45	0.78	0.79	4.39
	0.25	0.86	0.87	4.18	0.87	0.87	4.35	0.83	0.84	4.33
	0.50	0.90	0.91	4.16	0.88	0.89	4.20	0.87	0.88	4.26
	0.75	0.89	0.90	4.12	0.90	0.90	4.38	0.88	0.89	4.29
	1	0.91	0.92	4.15	0.91	0.92	4.37	0.92	0.93	4.35
floortile	0.05	0.77	0.11	0.24	0.77	0.07	0.26	0.77	0.08	0.24
	0.15	0.78	0.21	0.24	0.77	0.14	0.25	0.78	0.14	0.26
	0.25	0.78	0.19	0.26	0.77	0.16	0.23	0.77	0.13	0.26
	0.50	0.78	0.17	0.25	0.79	0.16	0.28	0.78	0.17	0.26
	0.75	0.79	0.17	0.27	0.79	0.14	0.26	0.78	0.14	0.24
	1	0.79	0.10	0.25	0.79	0.13	0.26	0.78	0.13	0.26
grid	0.05	0.96	0.59	5.47	0.96	0.84	5.64	0.96	0.50	5.50
	0.15	0.94	1.80	5.47	0.94	1.75	5.72	0.95	0.95	5.60
	0.25	0.93	2.18	5.38	0.91	2.64	5.69	0.94	1.80	5.62
	0.50	0.91	3.55	5.39	0.91	3.07	5.51	0.91	2.84	5.60
	0.75	0.89	3.91	5.53	0.91	3.25	5.72	0.91	3.23	5.64
	1	0.88	4.34	5.54	0.88	4.09	5.57	0.91	3.14	5.66
hanoi	0.05	0.85	1.01	1.18	0.85	1.11	1.14	0.88	0.59	1.14
	0.15	0.75	2.46	1.13	0.80	1.85	1.12	0.81	1.60	1.19
	0.25	0.74	2.48	1.11	0.76	2.45	1.14	0.77	2.00	1.15
	0.50	0.69	3.32	1.12	0.72	2.93	1.13	0.73	2.54	1.15
	0.75	0.70	2.97	1.10	0.70	2.83	1.12	0.71	2.98	1.12
	1	0.71	2.72	1.07	0.70	2.86	1.10	0.72	2.68	1.10
logistics	0.05	0.77	0.51	1.94	0.77	0.44	2.02	0.77	0.43	2.01
	0.15	0.72	0.89	1.98	0.75	0.78	2.01	0.77	0.74	2.02
	0.25	0.74	0.87	2.03	0.75	0.78	2.02	0.76	0.66	2.10
	0.50	0.77	0.74	1.94	0.76	0.76	2.03	0.77	0.66	1.98
	0.75	0.77	0.76	2.01	0.76	0.74	2.00	0.77	0.76	2.01
	1	0.79	0.68	1.96	0.77	0.75	2.01	0.77	0.73	2.06
openstacks	0.05	0.63	0.65	4.24	0.63	0.45	4.40	0.63	0.44	4.42
	0.15	0.74	1.02	4.25	0.74	0.90	4.41	0.71	0.84	4.52
	0.25	0.81	0.99	4.25	0.77	0.92	4.56	0.74	0.96	4.47
	0.50	0.83	0.80	4.29	0.77	0.74	4.46	0.78	0.77	4.62
	0.75	0.79	0.63	4.24	0.81	0.66	4.57	0.80	0.64	4.40
	1	0.76	0.53	4.24	0.77	0.55	4.49	0.75	0.54	4.50
pathways	0.05	0.95	0.20	0.74	0.94	0.46	0.76	0.95	0.15	0.75
	0.15	0.93	0.61	0.71	0.93	0.54	0.75	0.94	0.43	0.75
	0.25	0.93	0.82	0.73	0.92	0.85	0.75	0.93	0.54	0.73
	0.50	0.94	0.49	0.72	0.93	0.61	0.77	0.93	0.70	0.74
	0.75	0.94	0.49	0.72	0.94	0.32	0.76	0.94	0.41	0.77
	1	0.94	0.41	0.73	0.94	0.37	0.75	0.94	0.34	0.76
sokoban	0.05	0.69	0.40	3.95	0.68	0.36	4.07	0.68	0.40	4.02
	0.15	0.62	0.92	3.92	0.65	0.80	4.05	0.63	0.84	3.92
	0.25	0.62	0.98	4.07	0.60	1.14	4.02	0.62	1.08	3.88
	0.50	0.64	1.04	4.01	0.63	1.06	3.92	0.64	1.00	3.91
	0.75	0.65	0.96	3.99	0.61	0.96	4.04	0.64	1.03	4.10
	1	0.65	0.90	4.01	0.64	0.93	4.02	0.62	0.96	4.07
visitall	0.05	0.63	1.30	2.16	0.64	1.01	2.15	0.65	1.22	2.14
	0.15	0.51	2.28	2.26	0.56	2.09	2.13	0.61	1.80	2.29
	0.25	0.47	2.61	2.19	0.54	2.30	2.33	0.58	2.11	2.21
	0.50	0.47	2.66	2.10	0.50	2.52	2.24	0.54	2.34	2.29
	0.75	0.46	2.73	2.14	0.52	2.47	2.25	0.58	2.25	2.23
	1	0.48	2.68	2.19	0.56	2.38	2.30	0.61	2.17	2.30

(*continued on next page*)

Table 7 (*continued*).

Domain	Obs	Noise 0.0			Noise 0.1			Noise 0.2		
		Acc	%Spread	Time	Acc	%Spread	Time	Acc	%Spread	Time
zenotravel	0.05	0.87	0.23	0.45	0.87	0.21	0.47	0.87	0.33	0.46
	0.15	0.83	0.79	0.43	0.83	0.68	0.47	0.83	0.72	0.47
	0.25	0.82	0.89	0.45	0.82	0.82	0.49	0.82	0.88	0.44
	0.50	0.81	1.21	0.46	0.81	1.14	0.45	0.82	0.97	0.46
	0.75	0.82	1.04	0.45	0.83	1.02	0.45	0.82	1.09	0.45
	1	0.83	1.02	0.44	0.83	1.00	0.45	0.83	1.03	0.45
Average	0.05	0.77	0.61	3.38	0.77	0.54	3.46	0.77	0.57	3.46
	0.15	0.76	1.11	3.40	0.77	1.02	3.48	0.78	0.93	3.49
	0.25	0.77	1.22	3.39	0.77	1.15	3.51	0.77	1.08	3.47
	0.50	0.77	1.28	3.39	0.77	1.22	3.45	0.77	1.15	3.49
	0.75	0.77	1.26	3.40	0.77	1.19	3.52	0.78	1.14	3.48
	1	0.77	1.22	3.38	0.77	1.17	3.48	0.78	1.11	3.51
σ (std dev)	0.05	0.14	0.35	3.33	0.13	0.31	3.43	0.14	0.29	3.45
	0.15	0.12	0.70	3.34	0.11	0.59	3.45	0.11	0.46	3.45
	0.25	0.13	0.77	3.36	0.12	0.79	3.43	0.11	0.61	3.45
	0.50	0.14	1.12	3.40	0.13	0.96	3.40	0.12	0.85	3.44
	0.75	0.13	1.16	3.35	0.13	1.01	3.46	0.11	1.00	3.45
	1	0.13	1.24	3.31	0.12	1.17	3.43	0.11	0.94	3.46

Table 8
Accuracy, %spread and execution time (in seconds) for goal recognition for Π_{10} with different observability degrees (0.05...1) and noise (0.0, 0.1 and 0.2). The average and standard deviation (σ) results are shown in the last two blocks of rows.

Domain	Obs	Noise 0.0			Noise 0.1			Noise 0.2		
		Acc	%Spread	Time	Acc	%Spread	Time	Acc	%Spread	Time
blocksworld	0.05	0.82	0.37	3.02	0.81	0.54	3.17	0.82	0.49	3.23
	0.15	0.80	0.86	3.13	0.79	0.78	3.22	0.80	0.74	3.24
	0.25	0.78	1.11	3.09	0.78	1.14	3.20	0.78	0.92	3.21
	0.50	0.78	1.24	3.22	0.75	1.29	3.19	0.78	1.04	3.24
	0.75	0.78	1.28	3.11	0.80	1.01	3.26	0.79	1.08	3.30
	1	0.80	1.12	3.20	0.80	1.05	3.18	0.79	0.94	3.26
driverlog	0.05	0.73	0.40	6.31	0.74	0.35	6.56	0.75	0.35	6.50
	0.15	0.69	0.93	6.26	0.72	0.74	6.67	0.73	0.72	6.61
	0.25	0.67	1.15	6.07	0.68	1.07	6.52	0.69	1.01	6.60
	0.50	0.66	1.32	6.27	0.67	1.25	6.55	0.66	1.33	6.48
	0.75	0.64	1.54	6.40	0.64	1.44	6.58	0.64	1.46	6.55
	1	0.63	1.57	6.34	0.63	1.56	6.61	0.65	1.48	6.51
elevator	0.05	0.43	0.42	1.40	0.44	0.44	1.45	0.39	0.38	1.40
	0.15	0.71	0.74	1.36	0.66	0.70	1.38	0.66	0.70	1.40
	0.25	0.75	0.81	1.39	0.73	0.79	1.36	0.73	0.77	1.41
	0.50	0.80	0.86	1.38	0.76	0.81	1.43	0.77	0.81	1.38
	0.75	0.80	0.85	1.37	0.79	0.83	1.41	0.77	0.82	1.38
	1	0.82	0.87	1.36	0.81	0.86	1.42	0.79	0.84	1.40
floortile	0.05	0.77	0.07	0.11	0.76	0.10	0.11	0.76	0.10	0.11
	0.15	0.76	0.23	0.11	0.77	0.19	0.12	0.77	0.18	0.13
	0.25	0.74	0.31	0.11	0.77	0.17	0.11	0.75	0.17	0.11
	0.50	0.76	0.33	0.12	0.76	0.28	0.10	0.76	0.20	0.10
	0.75	0.76	0.22	0.11	0.76	0.26	0.11	0.76	0.22	0.12
	1	0.75	0.25	0.11	0.75	0.25	0.11	0.75	0.25	0.11
grid	0.05	0.97	0.40	3.84	0.96	0.84	3.86	0.96	0.84	3.75
	0.15	0.95	1.96	3.77	0.95	1.48	3.87	0.95	1.96	3.86
	0.25	0.94	2.52	3.74	0.94	2.56	3.73	0.94	2.68	3.72
	0.50	0.92	3.64	3.78	0.92	3.36	3.72	0.92	3.64	3.87
	0.75	0.90	4.52	3.75	0.91	3.96	3.88	0.93	3.40	3.91
	1	0.90	4.72	3.78	0.90	4.40	3.87	0.92	3.92	3.89
hanoi	0.05	0.85	0.67	0.16	0.86	0.57	0.15	0.87	0.47	0.14
	0.15	0.83	1.13	0.14	0.80	1.30	0.14	0.83	0.96	0.14
	0.25	0.79	1.51	0.13	0.77	1.86	0.17	0.81	1.13	0.16
	0.50	0.77	1.70	0.14	0.79	1.53	0.14	0.80	1.51	0.17
	0.75	0.78	1.59	0.14	0.77	1.61	0.14	0.77	1.57	0.15
	1	0.76	1.64	0.15	0.77	1.70	0.17	0.77	1.66	0.16
logistics	0.05	0.78	0.42	2.26	0.78	0.44	2.32	0.79	0.30	2.30
	0.15	0.76	0.78	2.25	0.79	0.63	2.33	0.78	0.57	2.31
	0.25	0.79	0.66	2.24	0.79	0.71	2.30	0.79	0.66	2.31
	0.50	0.80	0.64	2.31	0.80	0.61	2.27	0.81	0.60	2.31
	0.75	0.80	0.67	2.28	0.81	0.59	2.30	0.81	0.59	2.28
	1	0.81	0.62	2.25	0.81	0.64	2.28	0.81	0.56	2.30

(*continued on next page*)

Table 8 (*continued*).

Domain	Obs	Noise 0.0			Noise 0.1			Noise 0.2		
		Acc	%Spread	Time	Acc	%Spread	Time	Acc	%Spread	Time
openstacks	0.05	0.64	0.65	0.75	0.67	0.69	0.77	0.67	0.40	0.77
	0.15	0.62	1.15	0.74	0.64	1.08	0.74	0.62	1.11	0.78
	0.25	0.64	1.24	0.73	0.63	1.20	0.74	0.64	1.24	0.75
	0.50	0.68	1.12	0.75	0.66	1.17	0.76	0.65	1.05	0.73
	0.75	0.71	0.94	0.76	0.71	0.90	0.75	0.69	0.96	0.75
	1	0.71	0.94	0.73	0.71	0.96	0.81	0.70	0.92	0.74
pathways	0.05	0.96	4.14	0.18	0.97	4.00	0.19	0.96	3.71	0.20
	0.15	0.91	11.00	0.19	0.92	10.29	0.21	0.93	8.29	0.20
	0.25	0.89	13.71	0.20	0.91	11.43	0.20	0.92	10.14	0.22
	0.50	0.86	17.86	0.20	0.91	12.14	0.20	0.92	9.57	0.20
	0.75	0.89	14.57	0.21	0.90	12.57	0.19	0.88	15.14	0.20
	1	0.90	13.43	0.21	0.89	14.43	0.20	0.91	12.29	0.20
sokoban	0.05	0.69	0.38	4.00	0.67	0.53	4.06	0.65	0.48	4.07
	0.15	0.63	0.78	3.99	0.63	0.99	4.10	0.61	0.97	4.07
	0.25	0.63	1.08	4.03	0.61	1.18	4.03	0.63	1.08	4.02
	0.50	0.64	1.09	3.94	0.63	1.18	3.88	0.63	1.21	4.00
	0.75	0.65	1.09	3.98	0.61	1.14	4.04	0.63	1.20	3.93
	1	0.65	1.12	3.98	0.64	1.16	3.92	0.61	1.20	3.89
visitall	0.05	0.63	0.88	1.43	0.64	0.90	1.42	0.62	0.90	1.43
	0.15	0.60	1.76	1.44	0.59	1.66	1.48	0.57	1.66	1.49
	0.25	0.58	1.97	1.46	0.57	1.86	1.47	0.61	1.71	1.50
	0.50	0.54	2.17	1.43	0.60	1.93	1.50	0.62	1.79	1.49
	0.75	0.57	2.10	1.44	0.62	1.89	1.50	0.63	1.81	1.53
	1	0.60	2.05	1.40	0.63	1.92	1.48	0.65	1.83	1.52
zenotravel	0.05	0.89	0.37	0.35	0.89	0.23	0.35	0.89	0.30	0.38
	0.15	0.85	0.99	0.37	0.85	0.83	0.37	0.85	0.74	0.37
	0.25	0.81	1.48	0.35	0.80	1.65	0.37	0.81	1.32	0.35
	0.50	0.82	1.51	0.36	0.79	1.96	0.35	0.77	2.11	0.34
	0.75	0.79	2.10	0.35	0.79	2.04	0.35	0.79	2.07	0.35
	1	0.80	2.06	0.34	0.79	2.01	0.37	0.78	2.18	0.37
Average	0.05	0.76	0.55	1.99	0.77	0.57	2.03	0.76	0.52	2.02
	0.15	0.76	1.08	1.98	0.76	1.01	2.05	0.76	1.00	2.05
	0.25	0.75	1.24	1.96	0.75	1.20	2.02	0.76	1.11	2.03
	0.50	0.75	1.33	1.99	0.75	1.25	2.01	0.76	1.19	2.03
	0.75	0.76	1.32	1.99	0.76	1.23	2.04	0.76	1.22	2.04
	1	0.76	1.31	1.99	0.76	1.27	2.03	0.76	1.22	2.03
σ (std dev)	0.05	0.15	1.08	1.96	0.15	1.03	2.02	0.16	0.97	2.01
	0.15	0.11	2.91	1.95	0.11	2.73	2.05	0.12	2.18	2.04
	0.25	0.11	3.64	1.91	0.11	2.99	2.00	0.11	2.67	2.02
	0.50	0.11	4.82	1.95	0.10	3.20	2.00	0.10	2.52	2.01
	0.75	0.10	3.92	1.97	0.10	3.36	2.03	0.10	4.06	2.02
	1	0.10	3.62	1.97	0.09	3.88	2.02	0.10	3.28	2.00

situations and there are several reasons for this behavior. First, the use of random observations leads again to unexpected results, where less observability returns better accuracy and %spread than higher observability. Also, observations on common predicates are counted in many goals and hinder goal recognition. Second, if wrong plans are recognized in step 3, the candidate goals for goal recognition grow unnecessarily and make this task more volatile. Third, we have noted the use of causal graphs is less informative in domains where the goals to recognize are very independent, have little interaction, or belong to very different plans. For instance, let us consider the visitall domain, where there are many many places to visit and many different connections (recall that the initial state and goals do not need to be the same in the plan library). In such a case, the number of overlapping actions in the causal graphs tends to be more limited and less profitable.

If we focus on the standard deviation, the results for the accuracy are very good, as the deviation always remains below 0.15. The deviation of %spread and time show worse behavior because of the difference between domains, as it also happens in plan recognition.

Analogously to plan recognition, using noisy observations is not specially significant in goal recognition. The average section provides a clear picture here: the accuracy barely changes and the differences in the %spread are little. From the standard deviation perspective, the noise barely affects the accuracy and the time, whereas the changes in %spread are not very significant. This claim is also supported from a statistical perspective. We have calculated three ANOVAs (accuracy, %spread and time) with three populations each (noise 0.0, 0.1 and 0.2), and no differences exist in the population means with a confidence interval of 99%.

If we focus on Table 8, the results are very similar to Table 7. The differences in accuracy, on average, are 0.01–0.02 at most. Again, our approach does not need large plan libraries to get good accuracy in goal recognition, and Π_{10} seems sufficient. From a statistical standpoint, we have calculated the ANOVA for the accuracy with two populations (Π_{20} and Π_{10}) and there are no differences in the population means with a confidence interval of 99%. The tendencies for %spread and time are similar to Table 7, but the %spread and time for Π_{10} are smaller than in Π_{20} because the plan library is now smaller. The standard deviation is very good for the accuracy as well.

Using noisy observations is not problematic. In the table, we can see that the noise does not affect the standard deviation of the accuracy and time; noise only affects very slightly the %spread. The calculation of the three ANOVAs (accuracy, %spread and time) with three populations each (noise 0.0, 0.1 and 0.2) shows that there are no differences in the population means with a confidence interval of 99%.

As a conclusion for both tables, reaching an accuracy of 1 is highly unlikely because some real goals are missing in TP. This

also affects %spread because it shows less precision; and this happens with and without noise.

ROC analysis. We now graphically show the performance of the goal recognition task by means of ROC curves. Figs. 8 and 9 compare the TP *vs.* FP rates and TN *vs.* FN rates, respectively. We use a cloud of 24 points (12 of Π_{20} and 12 of Π_{10}) per observability degree. The optimal recognition is a point in the upper left corner and we want points above the diagonal.

When we focus on Fig. 8, the TP rate (recall) is clearly higher than the FP rate: most points have a FP rate below 0.35, although the TP rate is not particularly high either. Regarding Fig. 9, the TN rate (specificity) is higher than the FN rate: most points have a TN rate higher than 0.65, although the FN rate is also somewhat high. Two important consequences can be noted in both figures: noisy observations are not very significant, and almost all the points are above the diagonal, which is a desired result.

4.1.4. Discussion and lessons learned

We have learned several lessons, mainly from the point of view of the plan library (number and quality of the plans), the learned model and impact of the number of observations and noise:

- The plan library does not require optimal plans. The quality of the plans is not particularly relevant in our approach: we learn common structures based on the constraints imposed by the plans, no matter their quality. A straightforward consequence of our constraint-based approach is that it better learns what is impossible *w.r.t.* these constraints rather than what is conservatively possible. This means that, in some cases, our approach shows limitations to disambiguate plans or goals that might be possible.
- Having a library where the plans share few objects (e.g., blocks, floors, cities, trucks, etc.), where the initial/goal states are very different among plans, and where actions/goals have little interaction among them is a handicap, particularly in the detection of overlapping in causal graphs, which is key for goal recognition. All in all, the main limitation of using causal graphs in step 4 is similar to the limitation of landmark-extraction for goal recognition [13]. When there are many unrelated alternatives to support the goals, the odds of observing one of the landmarks are very low and the amount of overlapping in the causal graphs is drastically reduced.
- The size of the plan library has not a deep impact in the 3 + 4 recognition tasks, and the ANOVA tests confirm this in terms of the accuracy. Unlike other approaches, we do not need a large library, as it is not compulsory to learn the exact original planning model. It is sufficient to learn the essential relationships between actions. More particularly, once we have captured the essence of the planning model in step 1, having more plans is not crucial whatsoever. Our results for Π_{10} show the same tendencies of Π_{20}. The only limitation here is that we cannot recognize plans or goals that are not in the library.
- There is a subtle limitation in the learned model of step 1 *w.r.t.* negative predicates. Using classical PDDL domains and problems without negative preconditions/goals results in a model that ignores the negative effects, as they are unnecessary. In such a situation, the learned model always differs from the original one, as the negative information is not learned. This limitation can be addressed by simply including plans with negative preconditions or goals in the library, or by reasoning on mutex (mutually exclusive) constraints that infer negative effects [19]. However, the success of our approach does not depend on learning the exact original planning model.

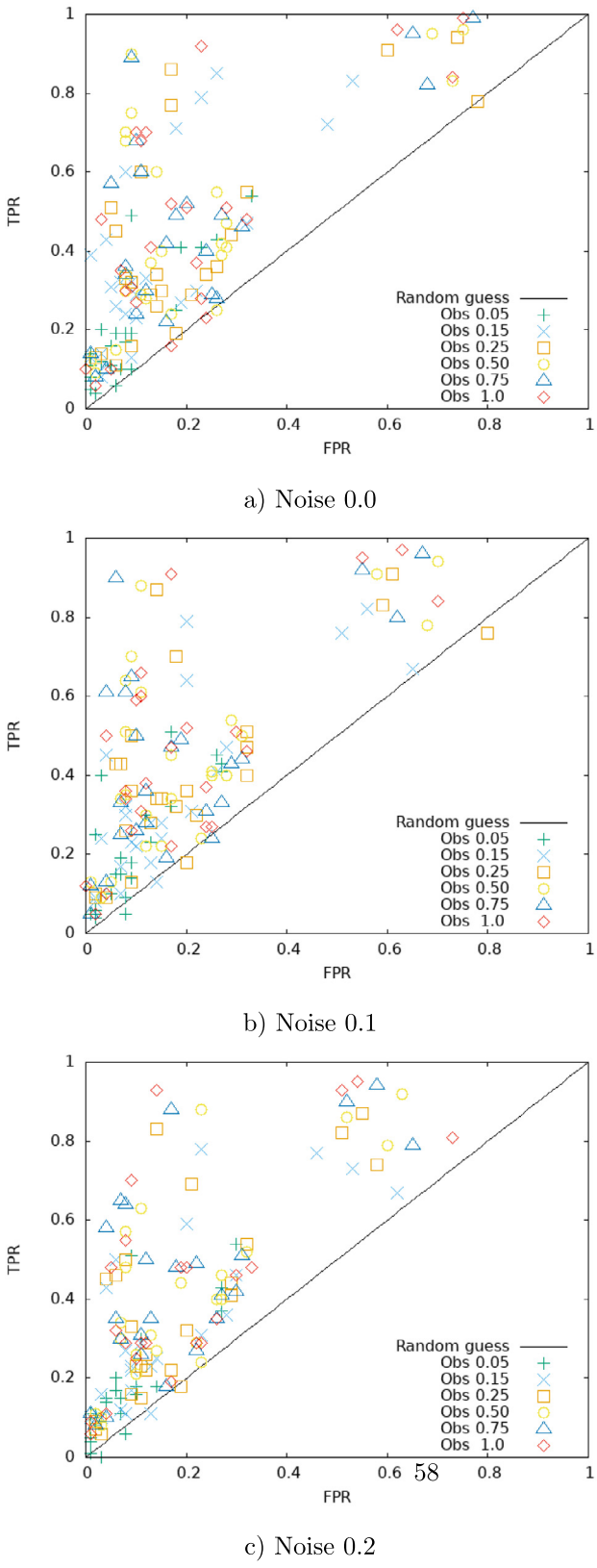
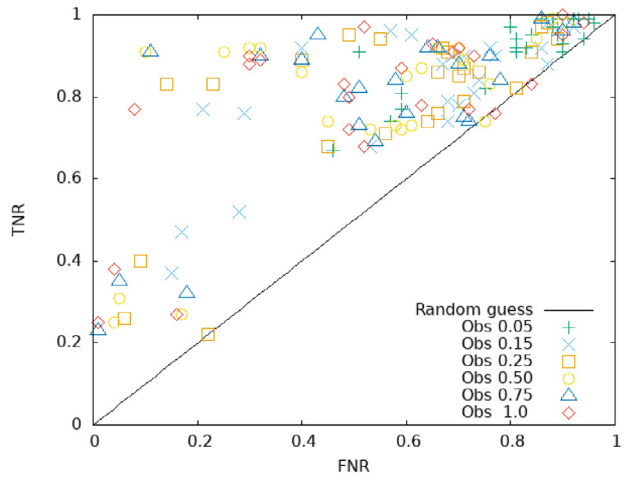
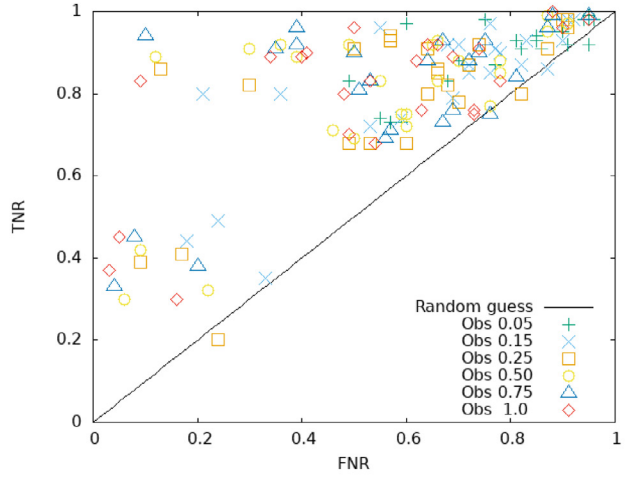


Fig. 8. Comparative ROC curve for goal recognition with different observability degrees: TP rate *vs.* FP rate with the three values of noise.

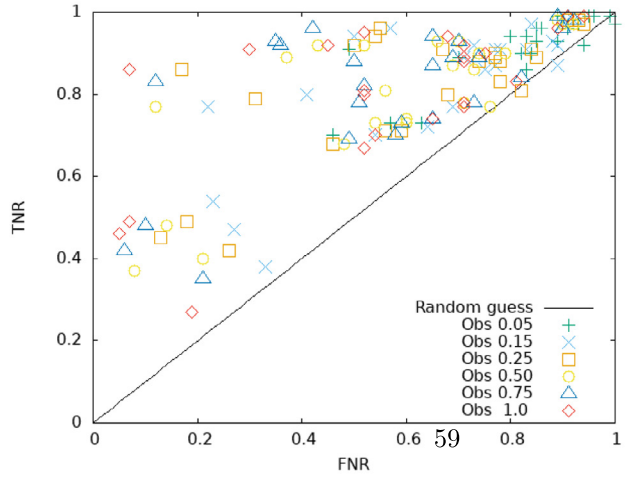
- To what extent does the approximate learned action model in step 1 represent the original model, and how does this



a) Noise 0.0



b) Noise 0.1



c) Noise 0.2

Fig. 9. Comparative ROC curve for goal recognition with different observability degrees: TN rate *vs.* FN rate with the three values of noise.

affect the recognition tasks? These two are interesting questions. To be answered, and to assess the importance of the learned model, we have repeated the experiments skipping the step 1. Rather than calculating an approximate solution $sol_{\mathcal{L}_n}$, we use the original operators in $\mathcal{O}$, which now include the negative effects. In the new experiments, the accuracy is nearly identical, with increments that do not exceed 0.06 and the differences in the spread are minimal. In other words, if the observations of step 2 only observe what it does happen (positive information) and not what it does not happen (negative information), the negative part of the action model is not essential. This means that the real difficulty of the recognition tasks relies on steps 3 and 4.

- Increasing the number of observations in step 2 means having a higher observability degree, which is more useful in plan recognition than in goal recognition. In plan recognition, a higher observability degree allows us to better discern among plans, thus providing better accuracy and spread (although there are some punctual exceptions because of the random observations). On average, the accuracy and spread monotonically improve when such a degree increases, both in Π_{20} and Π_{10}. In goal recognition, a higher observability degree improves the %spread, which is more precise both in Π_{20} and Π_{10}, but the accuracy remains more or less the same. In our approach, goal recognition is more complex and trickier than plan recognition, as the goals of the observed plan might not be the goals of one plan in the library, but a combination of goals of multiple plans. For instance, if we have to recognize goal g_1 and g_2, it might be possible that no plan in the library achieves both goals, so recognizing the right plan in step 3 is not vital. Consequently, our approach for goal recognition is less observation-dependent than plan recognition.
- If we just aim at goal recognition we might decide to skip the step 3. In order to assess this claim, we have repeated the goal recognition of step 4 without the plan recognition of step 3. The new results are just slightly worse, but the goal recognition task is now more costly because the set of candidate goals is much larger. That is, the accuracy follows the same tendency, but the execution time is increased. The main difficulty in goal recognition is to find a right threshold, and reducing the candidate goals thanks to the plans in $sol_{\mathcal{P}_\mathcal{R}}$ of step 3 proves useful to reduce the cost and the number of potential false positives in step 4. Therefore, skipping step 3 is not a fully convincing decision.
- Noisy observations do not significantly affect the success of the recognition tasks. We have not detected relevant differences when noise ranges from 0.0 to 0.2. In fact, our ANOVA tests show that, no matter this range of noise, there are no statistically differences with a confidence interval of 99%. The main reason is that a noisy observation that cannot be satisfied is simply not considered and does not affect the task.
- The execution time in all experiments is clearly affordable. Plan recognition depends more on the library size than goal recognition, but even in such a case the times do not exceed 15 s in Π_{20}.
- The ROC curves, in which we show the true–false trade-off for recall and specificity, provide good results (above the diagonal). The behavior is better in plan recognition than in goal recognition, although goal recognition is more tolerant to noise than plan recognition.

4.2. Comparison with other state-of-the-art model-based approaches

4.2.1. Setup

We now compare our approach with other model-based approaches. Since we do not have access to their binaries, we use

the results provided in [13] and their dataset[4] for goal and plan recognition. There are 15 domains in the dataset, but we can only parse 7 (the remaining domains do not use types or parameters, or use constants): blocksworld, depots, dwr, intrusion-detection, etc. The dataset contains thousands of experiments, which consist of a PDDL domain, an initial state, a set of candidate goals, a unique hidden goal in this set, and an observation sequence of actions that represent an optimal or sub-optimal plan that achieves the hidden goal. Five observability degrees are defined, 0.10, 0.30, 0.50, 0.70 and 1. A value of 0.10 means that 10% of the actions in the plan are observed. No predicates are observed in the dataset.

There is an additional version of the dataset to deal with noise. However, this version only includes two noisy observations per observation sequence that, on average, means that 12% of the observations are faulty [13]. In this version, there are only four observability degrees, 0.25, 0.50, 0.75 and 1.

Fig. 10 shows an input example of the dataset for blocksworld. According to the authors, the dataset is intended for goal and plan recognition. However, despite its name, it focuses essentially on plan recognition. As can be seen, there are 10 candidate goals $G_1, G_2 \ldots G_{10}$, but the recognition task only needs to recognize a unique hidden aggregate goal. In other words, the objective here is to recognize the atomic goal G_1, but no goals like *(clear D)*, *(on D R)* or *(on R A)* are individually recognized in the experiments provided in [13]. Consequently, this is equivalent to a plan recognition task $\mathcal{P}_{\mathcal{R}}$ with 10 candidate plans $\Pi_{10} = \{\pi_1, \pi_2 \ldots \pi_{10}\}$, which respectively achieve the goals $\{G_1, G_2 \ldots G_{10}\}$, where the optimal solution for the task is $sol_{\mathcal{P}_{\mathcal{R}}} = \{\pi_1\}$. Therefore, this comparison requires to use our plan recognition task.

The number of experiments depends on the domain and observability degree. For each experiment, we populate a plan library with the plans that achieve each candidate goal from the initial state. This is possible because the domain is part of the dataset. In the example of Fig. 10, we simply need to create a library with 10 candidate plans and the objective is to recognize the plan that achieves the hidden goal. Plans are solved by LPG.

4.2.2. Evaluated approaches and comparison

We compare our plan recognition task *vs.* some of the most successful approaches in literature:

- Ramirez & Geffner [4] (denoted as RG09), which uses heuristic estimators to approximate relaxed plans when the observations are included.
- Ramirez & Geffner [5] (denoted as RG10), which calculates the probability distribution over a set of goals, where the probability of the observations given a goal is defined in terms of the cost difference of achieving the goal when satisfying with the observations *vs.* not satisfying with them.
- E-Martin et al. [7] (denoted as ERS15), which propagates cost and interaction information in a plan graph, and uses this information to estimate goal probabilities over the candidate goals.
- Pereira et al. [13] (denoted as POM20), which defines two heuristic estimators by using landmarks. First, the goal completion heuristic h_{gc} estimates the percentage of completion of a goal based on the number of landmarks that have been detected and are required to achieve it. Second, the uniqueness heuristic h_{uniq} computes a value for every landmark and recognizes which candidate goal is being pursued from the observations. Both heuristics require a threshold to return the candidates with the highest estimate within that threshold. The authors use three values, 0, 10 and 20. In our comparison, we show the median threshold, i.e., 10.

[4] https://github.com/pucrs-automated-planning/goal-plan-recognition-dataset.

i) Domain:

See Figure 1

ii) Initial state:

D R A W O E P C - block

(handempty)
(clear O)
(ontable O)
(clear R)
(on R P)
(ontable P)
(clear E)
(ontable E)
(clear D)
(on D A)
(on A C)
(ontable C)
(clear W)
(ontable W)

iii) Candidate goals:

G_1: (clear D) (ontable W) (on D R) (on R A) (on A W)
G_2: (clear W) (ontable R) (on W A) (on A R)
G_3: (clear R) (ontable W) (on R A) (on A W)
G_4: (clear W) (ontable D) (on W A) (on A D)
G_5: (clear C) (ontable W) (on C R) (on R O) (on O W)
G_6: (clear R) (ontable W) (on R O) (on O W)
G_7: (clear W) (ontable R) (on W E) (on E A) (on A R)
G_8: (clear E) (ontable R) (on E A) (on A R)
G_9: (clear P) (ontable R) (on P E) (on E A) (on A R)
G_{10}: (clear R) (ontable E) (on R O) (on O P) (on P E)

iv) Hidden goal:

(clear D) (ontable W) (on D R) (on R A) (on A W)

v) Observation sequence of actions:

(unstack R P)

Fig. 10. Fragment of the dataset for blocksworld. The input for the recognition task includes: (i) the domain, as in Fig. 1; (ii) the initial state; (iii) the candidate goals; (iv) the hidden goal; and (v) the observations.

Table 9 depicts the results for the comparison when observations are noiseless. For each domain and observability degree, we present the number of experiments. Then, we present the average values for accuracy and spread for our approach and the five evaluated approaches. The optimal value for both accuracy and spread is 1. Since the computer used in [13] is faster than ours, a comparison of the execution times is unfair. For the interested reader, we also show the average solving time in seconds of our approach, which is now limited to 100 s for steps 1 + 2 + 3.

Our accuracy results are specially remarkable for low-medium observability degrees (up to 0.50–0.70). They are comparable to other approaches and even better in several domains, such as depots, ipc-grid, logistics and sokoban. When the plan is fully observed, the other approaches show better accuracy, except RG10. If we focus on the spread, our results are slightly worse on average, but a little better in depots and logistics.

Table 10 depicts the results for the comparison with noisy observations. Surprisingly, the results of other approaches for most domains are not provided in [13]. If we focus on intrusion-detection and ipc-grid, our accuracy for the intrusion-detection is better for low observability degrees but worse when the observability increases. In ipc-grid, our accuracy is clearly better, specially in comparison with RG09 and RG10. Our values for the spread are worse in intrusion-detection but better in ipc-grid.

If we compare Table 10 with the results of Table 9, we can see that including noise does not affect significantly to our accuracy, although the spread is a bit worse now. Summing up, our approach is sensibly more tolerant to noisy observations than the other approaches, specially with low-medium observability degrees. It also shows comparable results (better in some domains but worse in others), particularly in terms of accuracy and low observability degrees. This is very valuable when the observations are limited and/or noisy, taking into consideration that our approach does not need the planning action model as an input.

5. Conclusions

Recognition from observations is traditionally addressed by using large plan libraries or by using a planning model. The idea is to match observations over the plans, or match plans over the observations. Traditionally, authors that propose the use of a library claim that knowing a planning action model in advance is

Table 9
Comparison of our plan recognition task *vs.* other approaches in terms of accuracy and spread. Average results with different observability degrees and without noise.

Domain	Obs	Exp	Our $\mathcal{P}_{\mathcal{R}}$			RG09		RG10		ERS15		POM20-h_{gc}		POM20-h_{uniq}	
			Acc	Spread	Time	Acc	Spread	Acc	Spread	Acc	Spread	Acc	Spread	Acc	Spread
blocksworld	0.10	222	0.51	10.79	11.77	0.61	3.34	0.68	3.34	0.66	9.11	0.59	4.62	0.53	2.77
	0.30	234	0.72	6.59	30.15	0.76	1.79	0.47	1.84	0.78	10.53	0.79	3.96	0.67	2.51
	0.50	234	0.84	4.09	51.91	0.87	1.42	0.52	1.78	0.81	10.68	0.86	3.13	0.78	2.24
	0.70	234	0.91	2.69	83.10	0.95	1.18	0.59	1.67	0.90	8.63	0.96	2.51	0.91	2.02
	1	88	0.95	1.93	93.98	1.00	1.06	0.65	1.71	1.00	1.22	1.00	1.78	1.00	1.61
depots	0.10	77	0.78	2.74	9.05	0.67	2.89	0.50	1.61	[a]	[a]	0.67	3.42	0.49	2.70
	0.30	80	0.94	1.53	21.58	0.76	1.72	0.52	0.90	[a]	[a]	0.82	2.53	0.71	2.57
	0.50	81	0.97	1.26	37.37	0.89	1.33	0.50	0.75	[a]	[a]	0.94	1.77	0.87	2.10
	0.70	84	0.94	1.27	58.93	0.94	1.13	0.62	0.74	[a]	[a]	0.94	1.35	0.96	1.09
	1	28	0.94	1.07	73.78	1.00	1.07	0.75	0.72	[a]	[a]	1.00	1.05	1.00	1.09
dwr	0.10	42	0.26	2.62	13.73	0.80	2.29	0.56	1.63	0.93	6.38	0.86	4.32	0.70	3.07
	0.30	37	0.48	2.03	50.90	0.83	2.89	0.39	0.75	0.98	6.56	0.95	3.61	0.80	2.41
	0.50	29	0.74	1.52	63.71	0.86	1.64	0.41	0.62	0.99	6.27	0.98	3.19	0.88	2.16
	0.70	30	0.62	1.37	64.26	0.90	1.48	0.49	0.63	0.99	6.00	0.99	2.50	0.99	2.00
	1	8	0.83	1.50	92.85	0.93	1.14	0.57	0.64	1.00	1.00	1.00	1.67	1.00	1.60
intrusion-detection	0.10	105	0.82	4.02	4.02	1.00	2.53	0.81	2.37	0.90	3.18	0.96	2.56	1.00	2.54
	0.30	105	0.92	2.32	5.77	1.00	1.11	1.00	1.11	0.91	1.88	1.00	1.96	1.00	1.96
	0.50	105	0.92	2.17	7.14	1.00	1.00	1.00	1.02	0.94	1.45	1.00	1.19	1.00	1.91
	0.70	105	0.93	2.12	9.86	1.00	1.00	1.00	1.00	0.99	1.05	1.00	1.02	1.00	1.67
	1	45	0.93	2.11	12.58	1.00	1.00	1.00	1.00	1.00	1.04	1.00	1.02	1.00	1.60
ipc-grid	0.10	146	0.86	2.29	0.56	0.86	2.11	0.63	1.35	[a]	[a]	0.86	3.28	0.82	3.13
	0.30	153	0.95	1.37	1.45	0.88	1.25	0.81	0.90	[a]	[a]	0.88	2.32	0.90	2.34
	0.50	152	0.98	1.13	2.84	0.93	1.04	0.90	0.93	[a]	[a]	0.93	1.26	0.95	1.48
	0.70	153	0.98	1.07	3.38	0.97	1.00	0.95	0.95	[a]	[a]	0.97	1.15	0.97	1.16
	1	61	0.97	1.10	4.06	1.00	1.00	1.00	1.00	[a]	[a]	1.00	1.00	1.00	1.00
logistics	0.10	147	0.83	2.80	0.24	0.91	2.69	0.70	1.52	[a]	[a]	0.84	3.58	0.77	2.84
	0.30	153	0.97	1.30	0.62	0.97	1.35	0.83	0.99	[a]	[a]	0.95	2.14	0.87	1.96
	0.50	153	0.99	1.14	1.29	0.99	1.15	0.89	0.93	[a]	[a]	0.99	1.92	0.95	1.50
	0.70	153	0.99	1.05	2.33	1.00	1.11	0.92	0.94	[a]	[a]	0.99	1.39	0.99	1.47
	1	61	1.00	1.00	8.63	1.00	1.00	0.95	0.95	[a]	[a]	1.00	1.00	1.00	1.00
sokoban	0.10	15	0.77	2.60	0.77	0.71	2.51	0.71	0.85	0.68	2.98	0.87	2.89	0.68	2.78
	0.30	24	0.90	1.75	1.71	0.76	1.67	0.63	0.63	0.83	3.14	0.77	1.81	0.69	1.77
	0.50	17	0.94	1.35	3.05	0.86	1.63	0.68	0.74	0.82	2.27	0.88	1.80	0.83	1.80
	0.70	25	0.94	1.44	3.67	0.87	1.19	0.86	0.88	0.86	1.84	0.92	1.31	0.93	1.60
	1	11	0.85	1.73	4.40	1.00	1.03	0.96	0.96	0.86	1.03	1.00	1.00	1.00	1.03

[a]Means that no results are provided for that experiment.

difficult, and even impossible when the model is hidden. Authors that propose the use of an action model claim that having a plan library is costly and even impossible in many scenarios; although knowing the model allows us to indirectly populate plan libraries. Our approach places in between and uses a small library to approximate a (non-necessarily complete) action model, which is learned by a formulation that must satisfy the constraints imposed by the plans of the library. More concretely, we contribute with a unified 4-step approach for plan and goal recognition from unreliable observations that allows us to easily tune (and even skip) some individual steps.

Unlike typical approaches in literature, we clearly distinguish between plan and goal recognition. In plan recognition, the objective is to identify the observed plan. This is applicable when we want to identify what the agent is executing, specially in systems for intelligent detection, activity recognition and monitoring. In goal recognition, we move a step forward and the objective is to identify not necessarily one plan but individual goals that might belong to different plans. This is applicable when we want to identify what the agent expects to achieve, specially in proactive systems to foresee future scenarios. A key advantage of our approach is that we do not need an input of candidate goals, as they are filtered from the plan library, nor a large amount of observations to obtain good results.

Our approach has a strong practical application in scenarios with little information: domain theory unavailable, limited plan library, low observability degree and noisy observations. We are restricted to what we already know (previous plans) to automatically learn an action model. We get the most information out of that learned model and what we partially observe, by correctly ignoring noisy observations that cannot be satisfied, to recognize the plans that are consistent with the maximal number of observations. Then, we initialize the candidate goals from the recognized plans and exploit planning knowledge in the form of causal graphs and their overlapping to recognize the most promising goals. An interesting consequence is that the causal graphs are implicit in our unified constraint-based formulation.

Our formulation can be easily mapped into a SAT encoding. Although non-boolean variables make the SAT encoding more tedious, a positive result is that most of the constraints of our formulation can be turned into SAT clauses and managed by modern SAT solvers.

In absence of noise, the plan recognition task never under-recognizes, but when noise is considered it might under-/over-recognize, which also happens in goal recognition with and without noise. In other words, we cannot guarantee perfect recognition but, to our knowledge, no approach guarantees this. The comparison with other state-of-the-art approaches show that our results are comparable.

We acknowledge some limitations in our work so far. First, we are limited to the plans and goals of the plan library, so we cannot recognize new plans or goals (although this is common to most

Table 10
Comparison of our plan recognition task *vs.* other approaches in terms of accuracy and spread. Average results with different observability degrees and with two noisy observations (around 12% of noise).

Domain	Obs	Exp	Our $\mathcal{P}_{\mathcal{R}}$			RG09		RG10		ERS15		POM20-h_{gc}		POM20-h_{uniq}	
			Acc	Spread	Time	Acc	Spread	Acc	Spread	Acc	Spread	Acc	Spread	Acc	Spread
blocksworld	0.25	31	0.48	11.58	1.83	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	0.50	34	0.63	8.65	3.84	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	0.75	36	0.78	5.50	7.87	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	1	36	0.83	4.50	9.83	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
depots	0.25	34	0.60	4.59	3.33	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	0.50	35	0.83	2.43	8.43	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	0.75	36	0.96	1.25	18.51	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	1	36	0.95	1.39	22.19	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
dwr	0.25	23	0.68	3.26	29.96	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	0.50	24	0.75	2.58	66.49	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	0.75	24	0.81	1.75	78.31	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	1	24	0.85	1.42	89.23	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
intrusion-detection	0.25	90	0.83	3.73	3.81	0.63	2.34	0.39	0.81	0.43	2.31	0.69	4.43	0.59	3.94
	0.50	90	0.92	2.27	5.86	0.94	1.27	0.79	0.93	0.81	1.78	0.93	2.04	0.89	2.68
	0.75	90	0.93	2.11	8.56	0.99	1.01	0.93	0.94	0.93	1.10	0.99	1.33	0.94	1.182
	1	30	0.93	2.10	13.02	1.00	1.00	1.00	1.10	1.00	1.06	1.00	1.10	1.00	1.63
ipc-grid	0.25	84	0.90	1.90	0.19	0.11	2.81	0.10	1.00	[a]	[a]	0.76	2.95	0.76	2.83
	0.50	89	0.96	1.36	0.38	0.04	1.61	0.03	0.98	[a]	[a]	0.86	1.71	0.87	1.71
	0.75	90	0.99	1.12	0.68	0.08	1.10	0.08	0.92	[a]	[a]	0.94	1.23	0.94	1.15
	1	30	1.00	1.00	1.25	0.10	1.00	0.10	1.00	[a]	[a]	1.00	1.00	1.00	1.00
logistics	0.25	36	0.87	2.25	0.23	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	0.50	36	0.99	1.14	0.60	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	0.75	36	1.00	1.00	1.12	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	1	36	0.99	1.06	2.18	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
sokoban	0.25	19	0.69	3.47	1.11	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	0.50	19	0.90	1.79	2.48	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	0.75	14	0.92	1.57	4.54	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]
	1	16	0.96	1.25	5.30	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	[a]

[a]Means that no results are provided for that experiment.

recognition approaches). Second, our approach shows somewhat conservative: it is good to discard what cannot be recognized, as it violates the imposed constraints, but too permissive in recognition if no enough evidences can be extracted. Particularly in goal recognition, it can recognize two goals that are mutex (e.g., returning *(on S A)* and *(on R A)* in blocksworld). As part of future work, we want to include some kind of mutex reasoning to avoid contradictory goals and discover an ordering among them. Third, some careful tuning might help the recognition tasks. For instance, a large plan library could be detrimental for the success of the recognition tasks since the number of candidate plans/goals increases. However, tuning the plans in the library (to cover more situations and improve the model learned), or tuning the threshold in goal recognition (to better discern among goals) is interesting and part of our ongoing work. Finally, we recognize a set of plans/goals but we do not provide their score or probability distribution. A direct extension to reason on the existing information, which is future work, is to provide a ranking of probabilistic or otherwise hypotheses. We could give a weight to every observation, thus allowing a degree of confidence on it, and return a list of ranked plans and goals. For instance, we can extend the maximization problem of the plan recognition and the overlapping calculus between causal graphs of the goal recognition to deal with these weights.

CRediT authorship contribution statement

Antonio Garrido: Conception and design of study, Investigation, Software, Acquisition of data, Validation, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The link of the dataset is in the paper.

Acknowledgments

This work has been partially supported by grant PID2021-127647NB-C22 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe", and by the Spanish MINECO project TIN2017-88476-C2-1-R.

References

[1] D. Avrahami-Zilberbrand, G. Kaminka, Fast and complete symbolic plan recognition, in: Proc. International Joint Conference on AI (IJCAI-2005), 2005, pp. 653–658.
[2] P. Cohen, C. Perrault, J. Allen, Strategies for Natural Language Processing, Lawrence Erlbaum Associates, 1981, Ch. Beyond Question Answering.
[3] S. Jigui, Y. Minghao, Recognizing the agent's goals incrementally: planning graph as a basis, Front. Comput. Sci. China 1 (1) (2007) 26–36.
[4] M. Ramirez, H. Geffner, Plan recognition as planning, in: Proc. International Joint Conference on AI (IJCAI-2009), 2009, pp. 1778–1783.
[5] M. Ramirez, H. Geffner, Probabilistic plan recognition using off-the-shelf classical planners, in: Proc. of the AAAI Conference on Artificial Intelligence, 2010, pp. 1121–1126.
[6] S. Sohrabi, A. Riabov, O. Udrea, Plan recognition as planning revisited, in: Proc. International Joint Conference on AI (IJCAI-2016), 2016, pp. 3258–3264.

[7] Y. E-Martin, M. R-Moreno, D. Smith, A fast goal recognition technique based on interaction estimates, in: Proc. International Joint Conference on AI (IJCAI-2015), 2015, pp. 761–768.
[8] L. Santos, F. Meneguzzi, R. Pereira, A. Pereira, An LP-based approach for goal recognition as planning, in: Proc. of the AAAI Conference on Artificial Intelligence, 2021, pp. 11939–11946.
[9] G. Sukthankar, R. Goldman, C. Geib, D. Pynadath, H. Bui, Plan, Activity, and Intent Recognition. Theory and Practice, Morgan Kaufman, 2014.
[10] D. Pattison, D. Long, Domain independent goal recognition, in: Proc. of the Fifth Starting AI Researchers (STAIRS 2010), 2010, pp. 238–250.
[11] R. Goldman, C. Geib, C. Miller, A new model of plan recognition, in: Proc. of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI1999), 1999, pp. 245–254.
[12] H. Kautz, J. Allen, Generalized plan recognition, in: Proc. of the AAAI Conference on Artificial Intelligence, 1986, pp. 32–37.
[13] R. Pereira, N. Oren, F. Meneguzzi, Landmark-based approaches for goal recognition as planning, Artificial Intelligence 279 (2020) 1–49.
[14] A. Polyvyanyy, Z. Su, N. Lipovetzky, S. Sardina, Goal recognition using off-the-shelf process mining techniques, in: Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), 2020, pp. 1072–1080.
[15] M. Ramirez, H. Geffner, Goal recognition over POMDPs: inferring the intention of a POMDP agent, in: Proc. International Joint Conference on AI (IJCAI-2011), 2011, pp. 2009–2014.
[16] E. Charniak, R. Goldman, A bayesian model of plan recognition, Artificial Intelligence 64 (1993) 53–79.
[17] R. Mirsky, Y. Gal, S. Shieber, CRADLE: An online plan recognition algorithm for exploratory domains, ACM Trans. Intell. Syst. Technol. 8 (3) (2017) 1–45.
[18] J. Penberthy, D. Weld, UCPOP: a sound, complete, partial-order planner for ADL, in: Proc. Int. Conference on Principles of Knowledge Representation and Reasoning, Kaufmann, Los Altos, CA, 1992, pp. 103–114.
[19] A. Garrido, S. Jimenez, Learning temporal action models via constraint programming, in: Proc. of the 24th European Conference on Artificial Intelligence (ECAI 2020), 2020, pp. 2362–2369.
[20] A. Garrido, Learning temporal action models from multiple plans: A constraint satisfaction approach, Eng. Appl. Artif. Intell. 108 (2022) 104590.
[21] C. Geib, R. Goldman, A probabilistic plan recognition algorithm based on plan tree grammars, Artificial Intelligence 173 (2009) 1101–1132.
[22] H. Zhuo, L. Li, Multi-agent plan recognition with partial team traces and plan libraries, in: Proc. International Joint Conference on AI (IJCAI-2009), 2011, pp. 484–489.
[23] M. Ghallab, D. Nau, P. Traverso, Automated Planning. Theory and Practice, Morgan Kaufmann, 2004.
[24] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, D. Wilkins, PDDL - The Planning Domain Definition Language, AIPS-98 Planning Competition Committee, 1998.
[25] D. Aineto, S. Jiménez, E. Onaindia, Learning action models with minimal observability, Artificial Intelligence 275 (2019) 104–137.
[26] Q. Yang, K. Wu, Y. Jiang, Learning action models from plan examples using weighted MAX-SAT, Artificial Intelligence 171 (2–3) (2007) 107–143.
[27] M. Vilain, Getting serious about parsing plans: a grammatical analysis of plan recognition, in: Proc. of the AAAI Conference on Artificial Intelligence, 1990, pp. 190–197.
[28] D. Albrecht, I. Zukerman, A. Nicholson, Bayesian models for keyhole plan recognition in an adventure game, User Model. User-Adapt. Interact. 8 (1998) 5–47.
[29] B. Banerjee, L. Kraemer, Multi-agent plan recognition: Formalization and algorithms, in: Proc. of the AAAI Conference on Artificial Intelligence, 2010, pp. 1059–1064.
[30] N. Lesh, O. Etzioni, A sound and fast goal recognizer, in: Proc. Int. Joint Conference on AI (IJCAI-95), 1995, pp. 1704–1710.
[31] E. Onaindia, D. Aineto, S. Jimenez, A common framework for learning causality, Prog. Artif. Intell. 7 (2018) 351–357.
[32] S. Triantafillou, I. Tsamardinos, Constraint-based causal discovery from multiple interventions over overlapping variable sets, J. Mach. Learn. Res. 16 (2015) 2147–2205.
[33] T. Walsh, SAT v CSP, in: Proc. of the Int. Conference on Principles and Practice of Constraint Programming (CP-2000), 2000, pp. 441–456.
[34] A. Gerevini, A. Saetti, I. Serina, Planning through stochastic local search and temporal action graphs in LPG, J. Artificial Intelligence Res. 20 (2003) 239–290.