Plan Recognition as Probabilistic Trace Alignment

Jonghyeon Ko*, Fabrizio Maria Maggi*, Marco Montali*, Rafael Peñaloza†, Ramon Fraga Pereira‡
*Free University of Bozen-Bolzano, Italy
jongko@unibz.it, {maggi, montali}@inf.unibz.it
†University of Milano-Bicocca, Italy
rafael.penaloza@unimib.it
‡University of Manchester, United Kingdom
ramon.fragapereira@manchester.ac.uk

2023 5th International Conference on Process Mining (ICPM) | 979-8-3503-5839-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICPM60904.2023.10271943

***Abstract*—Plan Recognition is the task of identifying the goals and plans of an agent by observing its behavior within the environment. The problem has been extensively studied in the context of planning, in particular bringing forward stochastic techniques dealing with probability distributions over the possible agent goals, under the assumption that observations are reliable. More recently, a connection between this problem and process mining techniques has been established, paving the way towards the application of alignment-based conformance checking techniques from process mining to tackle plan recognition problems in a setting where observations may be faulty. In this work, we reconcile these two lines of research in a unified framework that deals at once with uncertainty over the goals and the faithfulness of observations. Instead of using *ad-hoc* techniques to solve this problem, we cast it as a probabilistic trace alignment problem, trading off between the similarity of observations and plans, and the likelihood that the agent is performing those plans. We assess the effectiveness of our approach by conducting a comparative experimental evaluation on state-of-the-art benchmarks.**

***Index Terms*—Trace Alignment, Stochastic Setting, Plan/Goal Recognition,**

I. Introduction

Plan Recognition is the task of identifying the plan (i.e., the sequence of actions) an observed agent is executing to achieve a particular goal by observing its interactions with an environment [1]. Such interactions can be seen as *observed events* on the behavior of an agent in its environment [2], or as actions performed by the agent (e.g., move, cook, drive), and state properties of the environment (e.g., at home, at work, resting). Recognition approaches take as input a recognition problem and return a set of plans that "best explain" an observed sequence Obs [2]. *Goal Recognition* is the problem of recognizing and returning the most likely goals an agent is aiming to achieve through the observed behavior. When the observed agent is not guaranteed to behave under perfect rationality (i.e., under optimality guarantees on the plan adopted to pursue its goals [3]), then it is not possible to precisely identify the executed plan. Instead, one can retrieve a set of likely plans which explain Obs, or compute the likelihood of the agent trying to pursue a given goal given Obs.

These problems have been investigated in two distinct, parallel lines of research. One line, developed within the planning community, deals with the inherent uncertainty about the goal that the agent intends to pursue. This problem translates into the uncertainty about the plan that is actually being executed. In this setting, not all executed actions may be observed, but all observations are reliable, i.e., one can complete them into a plan by plugging-in non-observed actions, without removing nor reordering the observed ones. In a second, parallel line of research, the approach in [4] has recently established a connection between a different form of goal recognition and *conformance checking* techniques in *Process Mining* [5], in particular those based on *alignments*. *Conformance checking* aims at comparing an actual, observed trace with a model capturing the set of legitimate traces; when the observed trace does not belong to such a set, alignments are used to measure how much the observations deviate from the expected behavior, by finding which are the *closest* legitimate trace(s) to the observed one. As shown in [5], this approach can be used to align possibly spurious (i.e., faulty or noisy) observations with the actual plans belonging to a reference action theory, relying on off-the-shelf trace alignment algorithms developed within the process mining community. A probability distribution is, in this case, inferred on the likelihood that the observed behavior matches with the possible actual plans, without considering any form of uncertainty on the agent goals.

In this paper, we reconcile these two lines of research: we *propose a unified framework that deals at once with uncertainty over the agent goals, and with faulty and noisy observations*. This interplay casts a plan recognition problem where two dimensions have to be simultaneously considered: (i) the similarity between observed behavior and plans, yielding a first probability measure on which plans likely match the observed actions, and (ii) the (indirect) likelihood of such plans in the light of which goals are most likely pursued, and which are less likely so. Indeed, it could be that the most similar plan is so unlikely as to be deemed almost impossible, yielding a suspect recognition. Conversely, the most likely plan may differ too much from the observation to be meaningful as a recognized plan. To suitably handle this trade-off, we infuse the approach in [3] with probabilistic trace alignment [6], which, in turn, can be practically solved by calling off-the-shelf trace aligners with distance measures that take into account probabilities. We assess the effectiveness of our approach by conducting an extensive experimental evaluation on well-known benchmarks, comparing the performance and

This research has been supported by the Free University of Bozen-Bolzano with the PRISMA project.

accuracy of our approach with the state-of-the-art recognition approaches in the literature; that is, planning-based and trace alignment-based recognition approaches.

II. Background and Formulation

We now recall the general definitions from plan recognition.

Definition 1. *A plan recognition problem is a tuple $\langle \mathcal{M}, Obs \rangle$, where $\mathcal{M}$ is a reference model describing possible behaviors of agents, and Obs is a sequence of observed events representing a plan performed by observed agents.*

To make the general *plan recognition problem* in Definition 1 more concrete, we ground it to the context of STRIPS [7], by following the definition of [8] and further refined in [3]. Notice that our approach seamlessly extends to richer planning settings such as PDDL [9], and, in general, it works for any planning domain whose execution semantics can be captured through finite-state transition systems.

We recall the necessary background from the literature and then introduce our extended setting, in which observations are potentially faulty, and the agent adopts a stochastic policy to select their next actions.

A. Planning Domains and Goal Libraries

We use [3], [8] as a basis to introduce STRIPS planning, with some reformulations that better suit our technical exposition without affecting the resulting framework.

Definition 2. *A (STRIPS)* planning domain *is a tuple $\mathcal{D} = \langle F, s_I, A \rangle$, where: F is a finite set of* fluents*; $s_I \subseteq F$ is the* initial state*; and A is a finite set of* actions*, where each action $a \in A$ is defined by a* positive precondition $Pre^+(a) \subseteq F$*, a* negative precondition $Pre^-(a)$*, an* add set $Add(a) \subseteq F$*, and a* delete set $Del(a) \subseteq F$.

A planning domain $\mathcal{D} = \langle F, s_I, A \rangle$ indicates how an agent can use actions in A to evolve a model of the world described using fluents in F, starting from the initial state s_I. Each state consists of a fluent evaluation, indicating which fluents from F are true therein. Action $a \in A$ is executable in state $s \subseteq F$ if all fluents in $Pre^+(a)$ belong to s and none of the fluents in $Pre^-(a)$ does. Executing a in s updates s by first inserting all fluents in $Add(a)$ and then removing those in $Del(a)$.

Technically, given a planning domain $\mathcal{D} = \langle F, s_I, A \rangle$, an action $a \in A$ is *executable* in state $s \subseteq F$ if $Pre^+(a) \subseteq s$ and $Pre^-(a) \cap s = \emptyset$. The action execution is defined as follows: state $s \subseteq F$ *results in* state $s' \subseteq F$ *through action* $a \in A$, written $s \xrightarrow{a} s'$, if a is executable in s, and $s' = (s \cup Add(a)) \setminus Del(a)$. In this case, there is a transition from s to s' through a. A state $s'' \subseteq F$ is *reachable* from s through actions in A, written $s \xrightarrow{A} s''$, if there exists a (possibly empty) sequence of transitions leading from s to s''. A sequence of transitions $\eta = s_0 \xrightarrow{a_0} \dots \xrightarrow{a_{n-1}} s_n$ induces a corresponding trace (i.e., action sequence) $\pi_\eta = a_0 \cdots a_{n-1}$ from A^*. For a trace $\pi = a_0 \cdots a_{n-1}$, we use the following notation: *(i)* ε denotes the empty trace; *(ii)* $|\pi| = n$ denotes the *length* of π; and *(iii)* for $i \in \{0, \dots, |\pi| - 1\}$, $\pi(i) = a_i$ extracts the action at position i in the trace, and $\pi^i = a_0 \cdots a_i$ extracts the prefix of π from the initial position to position i.

We now extend planning domains with multiple goals, and define what we call *goal (corpus) libraries* [10], representing equally valid alternative target goal states for the agent.

Definition 3. *A* goal library *is a tuple $\mathcal{L} = \langle F, s_I, A, \mathcal{G} \rangle$, where $\langle F, s_I, A \rangle$ is a planning domain, and $\mathcal{G}$ is a finite set of* possible goals*, where each goal $g \in \mathcal{G}$ comes with a* positive condition $Cond^+(g) \subseteq F$ *and* negative condition $Cond^-(g) \subseteq F$.

Slightly abusing the notation, given a planning domain $\mathcal{D} = \langle F, s_I, A \rangle$, we write $\langle \mathcal{D}, \mathcal{G} \rangle$ for the *goal library* $\langle F, s_I, A, \mathcal{G} \rangle$.

A *goal library* comes with a set of goals for the agent. Conceptually, the agent tries to achieve one of such goals, by attempting to execute an action sequence leading from s_I to a target *goal state* that *achieves* some goal $g \in \mathcal{G}$, that is, a state containing all fluents in $Cond^+(g)$ and none from $Cond^-(g)$.

For simplicity, we henceforth make the following three assumptions (which make sense when plan recognition is performed, like in our case, post-mortem):

1) *Goal disjointness*: goals in $\mathcal{G}$ are *pairwise disjoint*, that is, no state reachable from s_I through A achieves two distinct goals from $\mathcal{G}$;
2) *Traps as goals*: every reachable state from s_I through A that is a sink state (i.e., has no successor states) is actually a goal state;
3) *Stop at goals*: whenever the agent achieves one of the goals in the library, it stops.

The first assumption is used to map every goal state to its unique, achieved goal. This can be assumed without loss of generality: it is always possible (at the expense of introducing, in the worst case, exponentially many new goals) to turn an arbitrary goal library into one that satisfies goal disjointness. The second assumption is required to mark sink states as special states, interpreting, the corresponding goals not as conditions that the agent wants to achieve, but rather as conditions that the agent encounters when trapped. This assumption is also without loss of generality, as one can take an arbitrary goal library augmenting it with a special action that is executable only when no other action is executable (i.e., in the sink states of the original library), and that leads to a new sink state containing a special fluent indicating that the agent is trapped. The third condition conceptually captures that all goals in a library have equal importance. While the first two conditions are modeled within the goal library of interest, the third will be enforced when characterizing its execution semantics.

The execution semantics of a goal library is captured by a labeled transition system (LTS) whose states are subsets of fluents, and transitions are labeled by actions. The previous assumptions imply that every sink state in the LTS represents a final state associated to one and only one goal.

Definition 4. *An A-labeled LTS with goals in $\mathcal{G}$ is a tuple $\langle S, s_0, \Rightarrow, \gamma \rangle$, where S is a finite set of* states*, $s_0 \in S$ is the* initial state*, $\Rightarrow \subseteq S \times A \times S$ is an A-labeled transition relation;*

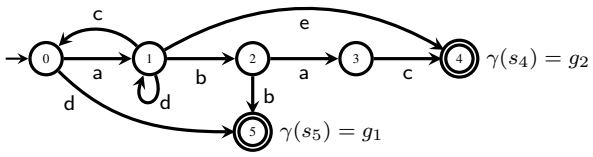

Fig. 1: Example of LTS induced by some goal library.

and $\gamma : S \to \mathcal{G}$ is a function mapping each sink state $s \in S$ *to a corresponding goal $\gamma(s) \in \mathcal{G}$.*

A goal library $\mathcal{L} = \langle \mathcal{D}, \mathcal{G} \rangle$ with $\mathcal{D} = \langle F, s_I, A \rangle$ induces an A-labeled LTS $\Upsilon_{\mathcal{L}} = \langle S, s_I, \Rightarrow, \gamma \rangle$ with goals in $\mathcal{G}$, where the initial state coincides with s_I, while $S \subseteq 2^F$, $\Rightarrow$ and γ are defined by mutual induction as the minimal sets satisfying the following conditions:

1) $s_I \in S$;
2) for every state $s \in S$,
   a) if $Cond^+(g) \subseteq s$ and $Cond^-(g) \cap s = \emptyset$ for some goal $g \in \mathcal{G}$, then γ is defined over s, and $\gamma(s) = g$;
   b) otherwise, for every action $a \in A$ executable in s, and state s' such that $s \xrightarrow{a} s'$, we have that $s' \in S$ and $\langle s, a, s' \rangle \in \Rightarrow$.

The set S coincides with the set of states reachable from s_I through actions in A, stopping at goal states. Notably, $\Upsilon_{\mathcal{L}}$ can be used to describe the set of *plan traces* leading from s_I to a goal state. Given $g \in \mathcal{G}$, a plan trace *achieves* g if it leads from s_I to a final state s such that $\gamma(s) = g$. We denote as $ptraces_g(\mathcal{L})$ the set of plan traces in $\mathcal{L}$ that achieves g. The overall set of plan traces is then $ptraces(\mathcal{L}) = \bigcup_{g \in \mathcal{G}} ptraces_g(\mathcal{L})$. Note that, due to the presence of loops, such a set is in general infinite; we do not prune loops as we do not place any assumption on how the agent is exploring the state space.

Example 1. *Fig. 1 depicts an LTS induced by some goal library $\mathcal{L}$ containing two goals g_1 and g_2 and five actions. Traces $\pi_1 =$* abb*, $\pi_2 =$* acd*, and $\pi_3 =$* abac *are all plan traces for $\Upsilon_{\mathcal{L}}$, the first two achieving g_1, the third g_2.*

B. Faulty and Noisy Observations

We now enter into our first contribution. We consider plan recognition problems over a domain planning model and a set of goals as in [8], with the difference that we consider an observed sequence modifiable; that is, a recorded action execution may correspond, in reality, to a different action, or to no action at all. The *closest* plan trace to a given observation then corresponds to the plan trace that *minimizes* the number of changes to transform the observations into such a trace.

This setting resembles that of *conformance checking via trace alignments* in process mining [5]. To formalize the problem at hand, we consequently recast the definitions of alignment and alignment costs from the process mining literature. When combining planning and trace alignment techniques, observations correspond to traces in an event log and plans to model paths. In addition, the analysis we perform is post-mortem, that is, the plan recognition is not performed at runtime but offline.

We rely on the distance formulation in [11]. Given a set A of actions and two traces $\sigma, \pi \in A^*$ respectively representing the *observed trace* recorded from the behavior of the agent and a *plan trace* extracted from the goal library, we use [11] to introduce a version of the edit distance that is parameterized by three functions:

- *Agent acting cost*, mapping A into $\mathbb{N} \cup \{\infty\}$ to capture the cost of deleting a given action from σ;
- *Model acting cost*, mapping A into $\mathbb{N} \cup \{\infty\}$ to capture the cost of adding a given action in σ;
- *Synchronous acting cost*, mapping $A \times A$ into $\mathbb{N} \cup \{\infty\}$ to capture the cost of replacing an action in σ into another.

Definition 5. *Consider an agent acting cost function ρ_A, a model acting cost function ρ_M, and a synchronous acting cost function $\rho_{||}$ over A. Let $\sigma \in A^*$ be an observed trace, $\pi \in A^*$ be a plan trace, and $i \in \{0, \ldots, |\sigma|\}$ and $j \in \{0, \ldots, |\pi|\}$ be positions in σ and π, respectively. The* alignment cost *between $\sigma|_i$ and $\pi|_i$ w.r.t. ρ_A, ρ_M, and $\rho_{||}$, written $\delta_{\rho_A, \rho_M, \rho_{||}}(\sigma, \pi)$ (and abbreviated $\delta(\sigma, \pi)$), is inductively defined as follows:*

$$\begin{aligned}
\delta(\varepsilon, \varepsilon) &= 0 \\
\delta(\sigma|_{i+1}, \varepsilon) &= \rho_A(\sigma(i)) + \delta(\sigma|_i, \varepsilon) \\
\delta(\varepsilon, \pi|_{i+1}) &= \rho_M(\pi(i)) + \delta(\varepsilon, \pi|_i) \\
\delta(\sigma|_{i+1}, \pi|_{j+1}) &= \min \begin{cases} \rho_{||}(\sigma(i), \pi(j)) + \delta(\sigma|_i, \pi|_j) \\ \rho_A(\sigma(i)) + \delta(\sigma|_i, \pi|_{j+1}) \\ \rho_M(\pi(j)) + \delta(\sigma|_{i+1}, \pi|_j) \end{cases}
\end{aligned}$$

The alignment cost between σ and π is $\delta(\sigma|_{|\sigma|}, \pi|_{|\pi|})$.

Example 2. *Consider the following* cost functions*: the agent and model acting cost functions assign a cost of 1 to each action, and the synchronous acting cost function yields a cost of 0 to change an action into itself, 1 otherwise. Then, $\delta(\sigma, \pi)$ returns the standard* Levenshtein distance *between σ and π. For the three plan traces of $\Upsilon_{\mathcal{L}}$ in Example 1 and the observed trace $\sigma =$* ab*, we get $\delta(\sigma, \pi_1) = 1$, $\delta(\sigma, \pi_2) = 2$, and $\delta(\sigma, \pi_3) = 2$.*

We are now ready to define the plan and goal recognition problem for goal libraries and faulty observations.

Definition 6. *A* recognition problem with faulty observations *is a tuple $\mathcal{P}_R = \langle \mathcal{L}, \sigma, \rho_A, \rho_M, \rho_{||} \rangle$, where $\mathcal{L} = \langle F, s_I, A, \mathcal{G} \rangle$ is a goal library; σ is an* observed trace *from A^*; and ρ_A, ρ_M, and $\rho_{||}$ are an agent acting cost function, a model acting cost function, and a synchronous acting cost function, respectively. A* solution *for $\mathcal{P}_R$ is a set $\Pi(\mathcal{P}_R)$ of plan traces from $ptraces(\mathcal{L})$ that are at the minimal alignment cost from σ:*

$$\Pi(\mathcal{P}_R) = \{\pi \in ptraces(\mathcal{L}) \mid \delta_{\rho_A, \rho_M, \rho_{||}}(\pi, \sigma) = \delta_{min} \neq \infty\},$$

where $\delta_{min} = \min\{\delta_{\rho_A, \rho_M, \rho_{||}}(\pi, \sigma) \mid \pi \in ptraces(\mathcal{L})\}$. The goal recognition solution *of $\mathcal{P}_R$ is the set*

$$\Gamma(\mathcal{P}_R) = \{g \in \mathcal{G} \mid \pi \text{ achieves } g \text{ for some } \pi \in \Pi(\mathcal{P}_R)\}.$$

Choosing the cost functions so that we only allow for adding actions in the observed trace (ρ_A always returns ∞, $\rho_{||}$ returns 0 for identical actions and ∞ otherwise, while ρ_M always returns 1), we reconstruct exactly the *subsequence distance* of [8], where observations are trustworthy, in the sense that they cannot be explained through plan traces that remove or replace the observed actions.

Example 3. *Consider the recognition problem $\mathcal{P}_R$ defined as follows: its goal library induces the LTS in Figure 1, it employs the Levenshtein distance, and its observed trace is $\sigma =$ ab. The two plan traces at minimum distance, thus forming the solution of $\mathcal{P}_R$, are* abb *(which extends σ with an additional action, achieving g_1) and* ae *(which considers the second action in σ as faulty, and replaces it with* e*, achieving g_2). If instead of the Levenshtein distance, $\mathcal{P}_R$ employs the subsequence distance, then its plan recognition solution is* abb*, and its goal recognition solution g_1.*

We now show how $\Pi(\mathcal{P}_R)$ can be computed in the case of subsequence and Levenshtein.

Subsequence. By definition of δ, $\Pi(\mathcal{P}_R)$ contains the *shortest* plan traces of $\mathcal{P}_R$ that have $\sigma = a_0 \cdots a_n$ as a subsequence.

We first show how we can compute a finite-state transition system whose traces are all and only the plan traces of $\mathcal{P}_R$ that have σ as subsequence. We:

1) take the goal library $\mathcal{D}$ of $\mathcal{P}_R$ and construct the A-labeled LTS $\Upsilon_{\mathcal{D}}$ as shown in Definition 6;
2) encode σ into a deterministic finite-state automaton $\mathcal{A}_\sigma$ accepting the language $(A \setminus \{a_0\})^*; a_0; \cdots; (A \setminus \{a_n\})^*; a_n; A^*$. This is the language that contains all the traces over A that contain σ as a subsequence; and
3) construct the cross-product $\Upsilon^\sigma_{\mathcal{D}}$ of $\Upsilon_{\mathcal{D}}$ and $\mathcal{A}_\sigma$ using standard techniques [12]. This transition system accepts all and only those plan traces (i.e., traces of $\Upsilon_{\mathcal{D}}$) that have σ as subsequence (i.e., are also traces of $\mathcal{A}_\sigma$).

$\Pi(\mathcal{P}_R)$ can then be directly extracted from $\Upsilon^\sigma_{\mathcal{D}}$ by returning the traces that correspond to the shortest paths in $\Upsilon^\sigma_{\mathcal{D}}$ connecting the initial to a final state [6].

Levenshtein. Let m be the length of the shortest plan trace(s) from $\mathcal{L}$, and let $n = |\sigma|$.

By recalling that, under Levenshtein distance, every "alignment move" costs 1, for every shortest plan trace π_s from $\mathcal{L}$, we get that $\delta(\sigma, \pi_s) \leq \delta^* = n+m$. In fact, the cost of aligning σ to π_s is dominated by the number of steps required to first consuming the whole content of σ without changing position in π_s, and then consuming the whole π_s.

As a consequence, for every $\pi \in \Pi(\mathcal{P}_R)$, we have that $\delta(\sigma, \pi) \leq \delta^*$; otherwise, instead of π, $\Pi(\mathcal{P}_R)$ would contain the shortest plan traces from $\mathcal{L}$ that are also at a minimum distance to π - which would in turn be not larger than δ^*, as shown above.

For every plan trace π from $\mathcal{L}$ such that $|\pi| > 2n + m$, we have that $\delta(\sigma, \pi) > \delta^*$. In fact, in the best case, n positions from π would perfectly align with σ, leaving the remaining $> n + m$ positions to consume from π without moving σ.

Putting everything together, we get that every plan trace from $\Pi(\mathcal{P}_R)$ must have a length that cannot exceed $2n + m$. It is then sufficient to traverse $\Upsilon_{\mathcal{D}}$ fetching all the finitely many plan traces whose length is $\leq 2n + m$, and then select from this set the plan traces that have the smallest Levenshtein distance to σ.

C. Goal Probabilities in Goal Libraries

We now go one step further, taking into account the fact that the agent may associate its goal library $\mathcal{L}$ with a prior probability distribution $\mathbb{P}(\mathcal{G})$ over the goals in the library. The posterior probability distribution $\mathbb{P}(\mathcal{G}|\sigma)$ obtained after considering the observation σ can be computed directly from the probabilities of reaching the different goal states in $\Upsilon_{\mathcal{L}}$ starting from σ. Following the approach presented in [3], we compute $\mathbb{P}(\mathcal{G}|\sigma)$ to find $\arg\max_{g \in \mathcal{G}} \mathbb{P}(g|\sigma)$, i.e., the most likely goal when given the observation sequence σ.

In a stochastic setting, i.e., when goals are associated to probabilities, using the Bayes' rule, we have $\mathbb{P}(g|\sigma) = \alpha \times \mathbb{P}(g) \times \mathbb{P}(\sigma|g)$, where α is a normalizing constant, $\mathbb{P}(g)$ is the probability of goal g as specified in the prior probability distribution over goals $\mathbb{P}(\mathcal{G})$, and $\mathbb{P}(\sigma|g)$ is the probability that σ is observed knowing that goal g has been achieved.

In a non-stochastic setting, the posterior probability distribution $\mathbb{P}(\mathcal{G}|\sigma)$ can be instead computed using the approach adopted by general plan recognition works like [3], [4], with the assumption that the probability of a plan is inversely proportional to its cost. This non-stochastic setting assumes that all goals and all plans belonging to a specific goal are equally likely, but a lower distance between an observation and a plan is linked to a higher probability of the plan. Using this assumption, following [3], [4], the distance between an observation σ and a goal g can be linked to $\mathbb{P}(g|\sigma)$ using a Boltzmann distribution as follows:

$$\mathbb{P}(g|\sigma) = \frac{e^{-\beta \times \mathsf{costdiff}(\sigma, g)}}{\sum_{g' \in \mathcal{G}} e^{-\beta \times \mathsf{costdiff}(\sigma, g')}}, \tag{1}$$

where the function costdiff is a distance metric to be defined, and β is a positive constant.

In the next section, we introduce our plan recognition approach that, differently from other plan recognition approaches like [3], [4], can be used both in a stochastic and in a non-stochastic setting (i.e., we assume that $\mathbb{P}(\mathcal{G})$ and $\mathbb{P}(\sigma|\mathcal{G})$ are not necessarily uniform). This characteristic becomes crucial when, as usually happens, a goal in a goal library is more often targeted than another, or a plan is more frequently observed than others for a specific goal.

III. Probabilistic Trace Alignment for Plan Recognition

From now on, we consider plan recognition problems with faulty observations.

A. Probabilistic Alignment of Plans and Observations

Given a recognition problem $\mathcal{P}_R = \langle \mathcal{L}, \sigma, \rho_A, \rho_M, \rho_{||} \rangle$, we describe how to relate observations to plans via *probabilistic trace alignment*, in the style of [6]. The approach takes as input an observed trace σ, a goal of interest g, and a set of plan traces $\pi \in ptraces_g(\mathcal{L})$ reaching g according to the goal library.[1] Then, a *ranking score* $\mathcal{R}(\sigma, \pi)$ is used to compute the plan recognition solution, which is the set of the plan traces having the highest ranking score:

$$\Pi(\mathcal{P}_R) = \{\pi \in ptraces(\mathcal{L}) \mid \mathcal{R}(\sigma, \pi) = \mathcal{R}_{max} \neq \infty\}.$$

As for the ranking score, we consider both the distance and the likelihood of a plan trace when recognizing the corresponding plan of a given observed trace. As discussed in the previous section, traditional trace alignment approaches [13], [14] consider the Levenshtein distance to produce aligned traces (i.e., the recognized plan traces). Instead, in our probabilistic approach, we extend it by computing the ranking score as $\mathcal{R}(\sigma, \pi) = \mathbb{P}(\pi|\sigma) \times s_\delta(\sigma, \pi)$, i.e., the product of the posterior probability of the plan trace when given the observed trace and the *similarity score* s_δ based on the Levenshtein distance — which measures the alignment cost.

As the posterior probability cannot be calculated when an observed trace does not conform to the LTS Υ_Q induced by the agent goal library, we introduce a penalty function that extracts a conforming prefix σ^i of the non-conforming trace σ, and multiplies its posterior probability by a penalty score $1/2^{[len(\sigma)-i-1]}$ as seen in Equation 2.

$$\mathbb{P}(\pi|\sigma) = \mathbb{P}(\pi|\sigma^i) \times \frac{1}{2^{[len(\sigma)-i-1]}}. \tag{2}$$

If there is no conforming prefix, the probability $\mathbb{P}(\pi|\sigma)$ is zero.

To compute the *similarity score* function $s_\delta(\sigma, \pi)$, instead, we first calculate the alignment cost $\delta(\sigma, \pi)$ using the Levenshtein distance, which is the sum of the agent acting costs, the model acting costs and the synchronous acting in the alignment. We then need to normalize the alignment cost in the probability space of the plan traces. One way to do so, adopted in [3], [4], is to use a Boltzmann distribution, yielding:

$$s_\delta(\sigma, \pi) = \frac{e^{-\beta \times \delta(\sigma,\pi)}}{\sum_{\pi' \in ptraces_g(\mathcal{L})} e^{-\beta \times \delta(\sigma,\pi')}}, \tag{3}$$

where, following [4], the confidence level β of the alignment is defined as:

$$\beta = \frac{1}{1 + \min_{\pi' \in ptraces_g(\mathcal{L})} \delta(\sigma, \pi')}. \tag{4}$$

Both the posterior probability $\mathbb{P}(\pi|\sigma)$ and the *similarity score* $s_\delta(\sigma, \pi)$ are probability mass distributions, i.e., $\mathbb{P}(\pi|\sigma), s_\delta(\sigma, \pi) \rightarrow [0, 1]$ and $\sum \mathbb{P}(\pi|\sigma) = 1, \sum s_\delta(\sigma, \pi) = 1$. Finally, $\arg\max_{\pi \in ptraces_g(\mathcal{L})} \mathcal{R}(\sigma, \pi)$ yields the optimal plan trace for an observed trace σ and a given goal g.

[1] Differently from standard trace alignment, here, plan traces do not correspond to model paths but to "happy paths" in a reference event log.

B. Relating Alignments to Goal Probabilities

To solve a goal recognition problem, i.e., to identify $\mathbb{P}(\mathcal{G}|\sigma)$ given an observation σ, first, we compute the set $\Pi(\mathcal{P}_R)$ of the optimal plan traces for all goals in $\mathcal{G}$ using probabilistic trace alignment. Then, we find the posterior probability distribution $\mathbb{P}(\mathcal{G}|\sigma)$ by computing, for each goal $g \in \mathcal{G}$, $\mathbb{P}(g|\sigma) = \alpha \times s_\delta(\sigma, \pi^*) \times \mathbb{P}(g) \times \mathbb{P}(\sigma|g)$, where $\pi^* \in \Pi(\mathcal{P}_R)$ is the optimal plan trace for g and $s_\delta(\sigma, \pi^*)$ is the same similarity function previously defined, i.e.:

$$s_\delta(\sigma, \pi^*) = \frac{e^{-\beta \times \delta(\sigma,\pi^*)}}{\sum_{\pi' \in \Pi(\mathcal{P}_R)} e^{-\beta \times \delta(\sigma,\pi')}}, \tag{5}$$

with the confidence level β defined as:

$$\beta = \frac{1}{1 + \min_{\pi' \in \Pi(\mathcal{P}_R)} \delta(\sigma, \pi')}. \tag{6}$$

Note that we can get $\delta(\sigma, \pi^*)$ without additional computations as this value was previously computed in the probabilistic trace alignment phase. As π^* represents the corresponding goal, a higher value of $s_\delta(\sigma, \pi^*)$ suggests that the goal corresponding to π^* is more likely.

Furthermore, in order to identify $\mathbb{P}(\mathcal{G}|\sigma)$, we need to compute the probability distributions $\mathbb{P}(\mathcal{G})$ and $\mathbb{P}(\sigma|\mathcal{G})$. If a prior probability distribution $\mathbb{P}(\mathcal{G})$ is not explicitly provided, it can be simply computed from its goal library using the frequency of plan traces. In particular, for each goal $g \in \mathcal{G}$, we have:

$$\mathbb{P}(g) = \frac{|\pi \in ptraces_g(\mathcal{L})|}{\sum_{g' \in \mathcal{G}} |\pi \in ptraces_{g'}(\mathcal{L})|}. \tag{7}$$

$\mathbb{P}(\sigma|\mathcal{G})$ is instead built by computing, for each goal $g \in \mathcal{G}$, $\mathbb{P}(\sigma|g)$ using a penalty function as follows:

$$\mathbb{P}(\sigma|g) = \mathbb{P}(\sigma^i|g) \times \frac{1}{2^{[len(\sigma)-i-1]}} \tag{8}$$

Finally, after the probability distribution $\mathbb{P}(\mathcal{G}|\sigma)$ has been computed, it is possible to identify the most likely goal through the formula $\max\arg_{g \in \mathcal{G}} \mathbb{P}(g|\sigma)$. The goal recognition approach just discussed is applicable both in a stochastic setting and in a non-stochastic setting where $\mathbb{P}(\mathcal{G})$ and $\mathbb{P}(\sigma|\mathcal{G})$ are assumed to be uniform.

IV. Experiments and Evaluation

In this section, we present a wide evaluation of the presented approach that is used to compare it against the most recent state-of-the-art approaches for goal recognition. We want to stress the fact that, differently from state-of-the-art approaches for plan recognition, we provide the user with a ranked list of top-K plans instead of identifying a single optimal plan. Therefore, although for comparing our approach with the existing ones we select the best alignment obtained, we are able to provide the user with a richer feedback.

Our evaluation aims at answering the research questions:

RQ1. Does the proposed approach improve the goal recognition accuracy of state-of-the-art approaches in a stochastic setting?

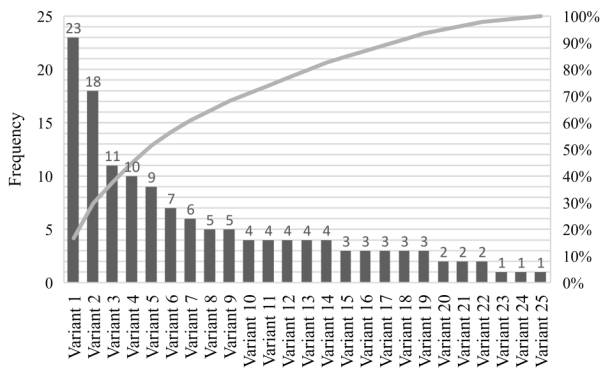

Fig. 2: Frequency distribution of plan traces for the subset '$goal0$' of DAILY LIVING.

Datasets	*# Goals*	*# Plan traces*	*# Variants*	*# Actions*
DAILY LIVING	8	27∼243	6∼43	14∼19
GRID NAVIGATION	9	47∼609	6∼102	12∼94
BLOCKS-WORLD	21	1000 (uniform)	1000	56∼94

TABLE I: Dataset statistics (variant indicates the trace type).

RQ2. Does the proposed approach accurately recognize goals in a non-stochastic setting when compared with state-of-the-art approaches?

Datasets and Experimental Setting. We consider a dataset taken from the *Process Mining* research field and two datasets taken from the *AI Planning* research field used in [4]. Regarding the process mining dataset, we use the DAILY LIVING[2] dataset, a real-world dataset that contains traces of daily living activities captured by sensors attached to 4 individuals. The traces are grouped by each individual and the parts of weekday/weekend, therefore, we achieve 8 groups of traces in total where each group is defined as a goal. For the planning domain datasets, we use GRID NAVIGATION[3] and BLOCKS-WORLD , which are typical datasets used in many *Classical Planning* research works and are based on PDDL [9], a generative modeling language for formalizing planning domains and problems. In particular, for GRID NAVIGATION, which is a plan recognition problem over grids, the model is learnt by traces in a setting of 10 for the size of the initial/final markings and 9 goals as it has been presented as the most challenging setting in [4].

For the stochastic setting, since the frequencies of the traces in all the datasets are uniform (i.e., for each trace π, $freq(\pi) = 1$), we replicated the traces, following an exponential distribution with rate parameter $\lambda = 1/5$, i.e., with the average frequency statistically converging to 5 (Fig. 2). For the BLOCKS-WORLD, we do not consider the stochastic setting because this data was already provided with a set of (crisp) PDDL models designed for a non-stochastic setting. We use this dataset to test our approach in a non-stochastic setting. TABLE I reports the statistics of the three datasets.

[2]https://doi.org/10.4121/uuid:01eaba9f-d3ed-4e04-9945-b8b302764176

[3]https://github.com/zihangs/fp1189_aamas2020/tree/master/datasets

Assumption	*Recall*
(1) $\mathbb{P}(g\|\sigma) = \mathbb{P}(g)$	0.313
(2) $\mathbb{P}(g\|\sigma) = s_\delta(\sigma, \pi)$	0.616
(3) $\mathbb{P}(g\|\sigma) = \alpha \times s_\delta(\sigma, \pi) \times \mathbb{P}(g)$	0.774
(4) $\mathbb{P}(g\|\sigma) = \alpha \times s_\delta(\sigma, \pi) \times \mathbb{P}(g) \times \mathbb{P}(\sigma\|g)$	0.932

TABLE II: *Recall* of the goal recognition task for different goal recognition schemas in the stochastic setting (using DAILY LIVING with 60%/40% split and $Obs(\%) = 30\%$).

For conducting our experiments, we split the datasets (DAILY LIVING and GRID NAVIGATION) into a train and a test set using ratio 60%/40% and then, to consider incomplete observations, we extract the prefix of the traces in the test set by considering only a certain percentage of events in each trace according to an observability degree, i.e., $Obs\ (\%) \in [10\%, 30\%, 50\%, 70\%, 100\%]$. Note that, by splitting the datasets into a train and a test set, it can happen that all the traces belonging to a certain variant fall into the test set. In this case, we have have faulty observations since those traces do not comply with any of the plan traces in the train set. For the BLOCKS-WORLD dataset, we have built a train set using 1000 plan traces generated from the provided PDDL models using a top-K planner [15].

In the evaluation, we use the performance metrics considered in [3], [4]. In particular, for each goal g, we try to recognize whether the goal the agent is trying to achieve is g (positive prediction), or is one of the goals in $\mathcal{G}\backslash\{g\}$ (negative prediction). Then, we compute the number of true positives TP, i.e., the number of correct positive predictions, the number of false negatives FN, i.e., the number of incorrect negative predictions, and the number of false positives FP, i.e., the number of incorrect positive predictions, and we calculate the three standard accuracy measures for the goal recognition problem: (i) *Precision* $= TP/(TP + FP)$ (ii) *Recall* $= TP/(TP + FN)$, and (iii) *F1-score* $= 2 \times (Precision \times Recall)/(Precision + Recall)$.

We have run the experiments on an Intel Core i9-7900X CPU with 3.3 GHz, 64GB RAM, running on MS Windows 10. All the datasets and the code for running the experiments are provided at https://github.com/jonghyeonk/PTA4PR.

Experimental Results in a Stochastic Setting. TABLE II shows how the goal recognition performance (in terms of *Recall*) achieved using different stochastic assumptions on the DAILY LIVING dataset. Assumption (1) is used to show the performance obtained using only the prior probability distribution over goals in the goal recognition task. Assumption (2) considers a non-stochastic setting where $\mathbb{P}(\mathcal{G})$ and $\mathbb{P}(\sigma|\mathcal{G})$ are uniform. When compared to assumption (2), the additions of $\mathbb{P}(\mathcal{G})$ and $\mathbb{P}(\sigma|\mathcal{G})$ provide a remarkable improvement of the recall as seen in the assumptions (3) and (4).

TABLE III shows the overall performance comparison between our approach (PTA) and the alignment-based approach [4], denoted as TA 2020. The table shows the average *Precision*, *Recall*, and execution time by varying the observability degree (*Obs %*) from 10% to 100%. For DAILY LIVING, our proposed approach shows higher *Precision* and *Recall* than TA 2020 regardless of the observability degree

Data	*Obs (%)*	*Approach*	*Precision*	*Recall*	*Time (min)*
DAILY LIVING	10	PTA	**0.776**	**0.592**	0.026
		TA 2020	0.331	0.058	0.025
	30	PTA	**0.972**	**0.932**	0.048
		TA 2020	0.467	0.129	0.036
	50	PTA	**0.977**	**0.934**	0.072
		TA 2020	0.656	0.184	0.045
	70	PTA	**0.983**	**0.952**	0.094
		TA 2020	0.528	0.253	0.052
	100	PTA	0.977	**0.932**	0.128
		TA 2020	**0.981**	0.372	0.053
GRID NAVIGATION	10	PTA	0.168	**0.269**	0.094
		TA 2020	**0.776**	0.238	0.064
	30	PTA	0.595	**0.559**	0.119
		TA 2020	**0.887**	0.502	0.072
	50	PTA	0.748	**0.745**	0.141
		TA 2020	**0.998**	0.655	0.078
	70	PTA	0.980	**0.956**	0.174
		TA 2020	**0.996**	0.703	0.077
	100	PTA	0.989	**0.960**	0.199
		TA 2020	**0.996**	0.769	0.075

TABLE III: Results by observability level (stochastic setting).

Data	*Obs (%)*	*Approach*	*Precision*	*Recall*	*Time (min)*
DAILY LIVING	10	PTA	**0.701**	**0.734**	0.017
		TA 2020	0.160	0.084	0.017
	30	PTA	**0.993**	**0.984**	0.036
		TA 2020	0.409	0.261	0.022
	50	PTA	**1.000**	**1.000**	0.053
		TA 2020	0.563	0.342	0.029
	70	PTA	**1.000**	**1.000**	0.070
		TA 2020	0.868	0.706	0.032
	100	PTA	**1.000**	**1.000**	0.091
		TA 2020	0.964	0.891	0.038
GRID NAVIGATION	10	PTA	0.190	0.233	0.042
		TA 2020	**0.667**	**0.216**	0.046
	30	PTA	0.376	**0.446**	0.055
		TA 2020	**0.614**	0.387	0.051
	50	PTA	0.402	0.308	0.073
		TA 2020	**0.581**	**0.530**	0.054
	70	PTA	0.468	0.298	0.085
		TA 2020	**0.587**	**0.594**	0.053
	100	PTA	0.452	0.416	0.099
		TA 2020	**0.588**	**0.618**	0.051

TABLE IV: Results by observability level (non-stochastic setting).

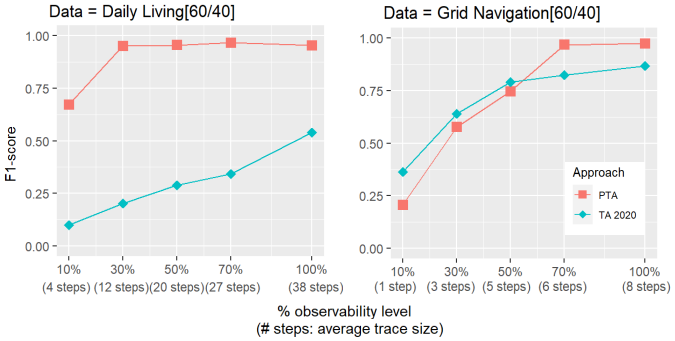

Fig. 3: *F1-score* results in a stochastic setting.

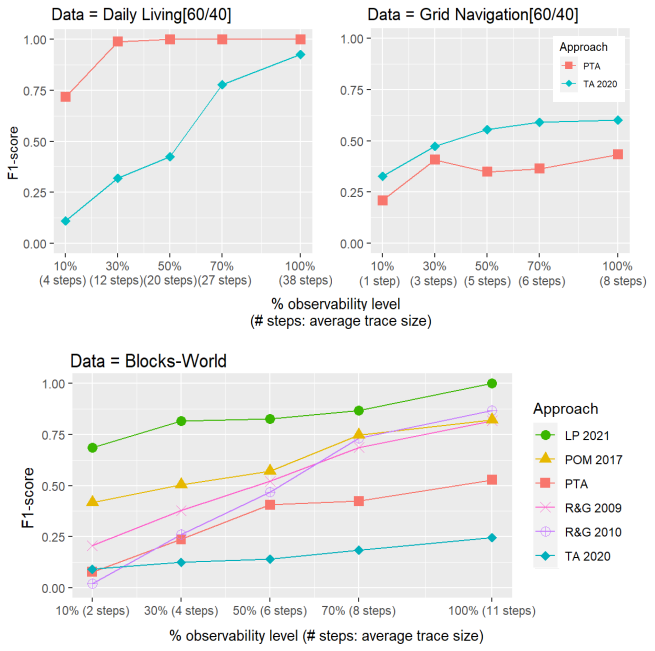

Fig. 4: *F1-score* by observability level (non-stochastic setting).

as, in this case, the prior probability distribution over goals helps the correct recognition of the goals. However, for GRID NAVIGATION, our approach does not perform equally well. This is due to the fact that the distribution of the trace lengths in GRID NAVIGATION is limited. The prior probability distributions are not significant enough to disambiguate among the goals, especially for low degrees of observability (e.g., 10% and 30%). This trend can also be seen in the *F1-score* in Figure 3. In terms of execution time, our approach takes longer because we calculate the alignment cost of all plan traces to provide a ranked list of alignments, whereas TA 2020 produces a single optimal alignment.

Note that, we do not adapt any of the planning-based approaches as baselines here because the train/test split can generate non-complying observations to plan traces. Next, we show the results and comparison against the planning-based approaches in the non-stochastic setting using the BLOCKS-WORLD dataset. Tha above discussion answers **RQ1**. In the remaining part of the section, we answer **RQ2**.

Experimental Results in a Non-Stochastic Setting. We now show the results in a non-stochastic setting. For this setting, we use DAILY LIVING and GRID NAVIGATION without replicated traces. Table IV shows the average *Precision*, *Recall*, and execution time for the two datasets. Also in this case, for DAILY LIVING, our approach shows good performance while, for GRID NAVIGATION, it shows worse performance when compared to the results obtained in the stochastic setting (Table III). This means that, for this dataset, the prior probability distribution over goals positively affects the performance.

Comparison with planning-based approaches. Figure 4 shows the results obtained by comparing our approach (PTA) with TA 2020, and four state-of-the-art planning-based approaches for goal recognition, i.e., (i) R&G 2009 [8], (ii) R&G 2010 [3] (Fast-Downward [16], A* with the LM -Cut heuristic [17]), (iii) POM 2017 [18] (*goal completion* heuristic) and (iv) LP 2021 [19], using the BLOCKS-WORLD dataset in a non-stochastic setting. In particular, we have tested the planning-based approaches using the PDDL models provided with the dataset, while our proposed approach and TA 2020 were tested using 1000 plan traces generated from the PDDL models using a top-K planner [15]. This means that the experi-

mental settings of the process mining-based approaches (PTA, TA 2020) and the planning-based approaches (R&G 2009, R&G 2010, POM 2017, LP 2021) are slightly different since the process mining-based approaches are not supported by a (correct and complete) domain model. We can see that LP 2021 outperforms all approaches for all metrics, including our approach. Overall, the planning-based approaches that rely on complete and correct PDDL domain models have higher performance than the process mining-based approaches. We note that formalizing complete and correct PDDL domain models are laborious, as it may require domain expertise, resulting in additional domain design effort, which is sometimes a limitation for these approaches. However, when looking at the approaches that use the incomplete domain information, we can see that our approach outperforms TA 2020 over all the observability degrees. We also note that our approach is competitive with the planning-based approaches (apart from LP 2021) for low observability degrees (10%, 30%, and 50%).

V. Related Work

We follow the Ramírez and Geffner's [3], [8] formalism to define the plan recognition problem, and adopt and extend their probabilistic framework to produce a set of optimal alignments, by trading off the similarity between observations and plans, and the likelihood that the agent is performing those plans. An approach similar to [3], [8] has been presented in [20], which extends the probabilistic framework of Ramírez and Geffner by reasoning over top-K plans. Existing approaches for probabilistic trace alignment [21], [22] have also proposed likelihood-based cost functions. The above approaches, differently from ours, do not consider that the recorded observations about the agent behavior are not only incomplete with missing observations (partial observability), but they may be faulty/noisy.

Polyvyanyy et al. [4] have demonstrated how to use off-the-shelf process mining techniques for recognizing goals in different domains. We provide two main contributions with respect to [4]. First, we provide a formal definition of the plan recognition problem with faulty observations. In addition, we consider the case in which the prior probability distributions over goals is not uniform, which is a very common case in many domains and we use probabilistic trace alignment as a possible solution to solve these recognition problems.

VI. Conclusions and Future Work

We have shown that plan recognition with faulty observations and goal probabilities can be effectively solved by using *probabilistic trace alignment* techniques. We have developed an approach that is able to rank the possible intended plans from observed traces by multiplying the similarity (based on the Levenshtein distance) between an observed trace and a plan trace by the probability of the plan trace.

As future work, we want to extend our approach to different forms of stochastic planning domains. First, we want to recast our approach to the case where the agent adopts a stochastic policy to choose its next action - yielding a Markov chain from its goal library. Second, we intend to consider actions with nondeterminism, leading to MPDs and/or POMDPs planning domains [23]. Finally, we plan to improve the execution times needed to compute the recognized plans by implementing approaches that do not return an optimal ranking of alignments but an approximate one, like the one presented in [6].

References

[1] C. F. Schmidt, N. S. Sridharan, and J. L. Goodson, "The plan recognition problem: An intersection of psychology and artificial intelligence," *Artificial Intelligence*, vol. 11, no. 1-2, pp. 45–83, 1978.

[2] R. Mirsky, S. Keren, and C. W. Geib, *Introduction to Symbolic Plan and Goal Recognition*. Morgan & Claypool Publishers, 2021.

[3] M. Ramírez and H. Geffner, "Probabilistic plan recognition using off-the-shelf classical planners," in *AAAI*, 2010.

[4] A. Polyvyanyy, Z. Su, N. Lipovetzky, and S. Sardina, "Goal recognition using off-the-shelf process mining techniques," in *AAMAS*, 2020.

[5] J. Carmona, B. F. van Dongen, A. Solti, and M. Weidlich, *Conformance Checking - Relating Processes and Models*. Springer, 2018.

[6] G. Bergami, F. M. Maggi, M. Montali, and R. Peñaloza, "Probabilistic trace alignment," in *ICPM*, 2021.

[7] R. Fikes and N. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, 1971.

[8] M. Ramírez and H. Geffner, "Plan recognition as planning," in *IJCAI*, 2009.

[9] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL − The Planning Domain Definition Language," in *AIPS*, 1998.

[10] N. Blaylock and J. F. Allen, "Corpus-based, statistical goal recognition," in *IJCAI*, G. Gottlob and T. Walsh, Eds., 2003.

[11] P. Felli, A. Gianola, M. Montali, A. Rivkin, and S. Winkler, "Cocomot: Conformance checking of multi-perspective processes via SMT," in *BPM*, 2021.

[12] W. M. van der Aalst, V. Rubin, B. F. van Dongen, E. Kindler, and C. W. Günther, "Process mining: A two-step approach using transition systems and regions," *BPM Center Report*, vol. 6, 2006.

[13] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance checking using cost-based fitness analysis," in *EDOC*, 2011.

[14] M. de Leoni and A. Marrella, "Aligning real process executions and prescriptive process models through automated planning," *Expert Systems with Applications*, vol. 82, 2017.

[15] M. Katz, S. Sohrabi, O. Udrea, and D. Winterer, "A novel iterative approach to top-k planning," in *ICAPS*, 2018.

[16] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[17] M. Helmert and C. Domshlak, "Landmarks, Critical Paths and Abstractions: What's the Difference Anyway?" in *ICAPS*, Oct. 2009.

[18] R. Pereira, N. Oren, and F. Meneguzzi, "Landmark-based heuristics for goal recognition," in *AAAI*, vol. 31, 2017.

[19] L. R. d. A. Santos, F. Meneguzzi, R. F. Pereira, and A. Pereira, "An LP-Based Approach for Goal Recognition as Planning," in *AAAI*, 2021.

[20] S. Sohrabi, A. V. Riabov, and O. Udrea, "Plan Recognition as Planning Revisited," in *IJCAI*, 2016.

[21] E. Bogdanov, I. Cohen, and A. Gal, "Conformance checking over stochastically known logs," in *BPM Forum*, 2022, pp. 105–119.

[22] M. Pegoraro, M. S. Uysal, and W. M. P. van der Aalst, "Conformance checking over uncertain event data," *Inf. Syst.*, vol. 102, p. 101810, 2021.

[23] M. Ramírez and H. Geffner, "Goal recognition over pomdps: Inferring the intention of a POMDP agent," in *IJCAI*, 2011.